河南大學

Henan University

I'm dying to play GTA 6

Contestant

范恒嘉 **Hengjia Fan**

张帆 **Fan Zhang**

宋明贤 **Mingxian Song**

目录

1 数据结构

1.1 DSU

```cpp
struct DSU
{
    vector<int> f, siz;

    DSU(int n) : f(n + 1), siz(n + 1, 1)
    {
        iota(f.begin(), f.end(), 0);
    }

    int find(int x)
    {
        if(x != f[x]) f[x] = find(f[x]);
        return f[x];
    }

    bool merge(int x, int y)
    {
        x = find(x);
        y = find(y);

        if (x == y)
            return false;

        if (siz[x] < siz[y])
            swap(x, y);

        siz[x] += siz[y];
        f[y] = x;
        return true;
    }

    int size(int x)
    {
        return siz[find(x)];
    }

    bool connected(int x, int y)
    {
        return find(x) == find(y);
    }
};
```

1.2 Fenwick

```cpp
template <typename T>
struct Fenwick {
    int n;
    std::vector<T> a;

    Fenwick(int n) : n(n), a(n + 1) {}


    void add(int x, const T &v) {
        for (int i = x; i <= n; i += i & -i) {
            a[i] = a[i] + v;
        }
    }

    T sum(int x) {
        T ans{};
        for (int i = x; i > 0; i -= i & -i) {
            ans = ans + a[i];
        }
        return ans;
    }

    T range(int l, int r) {
        return sum(r) - sum(l - 1);
    }

    T kth(T k) {
        int pos = 0;
        int logn = std::bit_width(a.size() - 1);
        for(int i = 1 << (logn - 1); i > 0; i >>= 1) {
            if(pos + i < a.size() && a[pos + i] < k) {
                k -= a[pos + i];
                pos += i;
            }
        }
        return pos + 1;
    }

};
```

1.3 ST 表

```cpp
template <typename T>
class SparseTable {

    // using func_type = function<T(const T &, const T &)>;
```

```
    vector<vector<T>> ST;
    // 更改操作，修改这一条语句
    static T op(const T &t1, const T &t2) { return max(t1, t2); }

    // func_type op;

    public:
    SparseTable(const vector<T> &v) {

        int n = v.size() - 1;

        int len = __lg(n);
        ST.assign(n + 1, vector<T>(len + 1, 0));

        for (int i = 0; i <= n; ++i) {
            ST[i][0] = v[i];
        }
        for (int j = 1; j <= len; ++j) {
            int pj = (1 << (j - 1));
            for (int i = 0; i + pj <= n; ++i) {
                ST[i][j] = op(ST[i][j - 1], ST[i + (1 << (j - 1))][j - 1]);
            }
        }
    }

    T query(int l, int r) {
        int q = __lg(r - l + 1);
        return op(ST[l][q], ST[r - (1 << q) + 1][q]);
    }
};
```

1.4 trie

```

struct Trie {

    int tot;
    vector<vector<int>> nex;
    vector<int> cnt; // 以这个节点结尾的字符串的个数
    Trie() : nex(100001, vector<int>(26)), cnt(100001) {}
    Trie(int n, int m) : nex(n + 1, vector<int>(m)), cnt(n + 1) {}
    // n为树中最多会有多少个节点，m表示有多少种字符
    void insert(string s, int t = 1) {    // 插入字符串
        int p = 0;
        for (int i = 0; i < s.length(); i++) {
            int c = s[i] - 'a';
            if (!nex[p][c]) nex[p][c] = ++tot;    // 如果没有，就添加结点
            p = nex[p][c];
        }
        cnt[p] += t;
    }

    bool find(string s) {    // 查找字符串
        int p = 0;
        for (int i = 0; i < s.length(); i++) {
            int c = s[i] - 'a';
            if (!nex[p][c]) return 0;
            p = nex[p][c];
        }
        return cnt[p];
    }
};
```

1.5 动态开点线段树

```
#include<bits/stdc++.h>
using namespace std;
using i64 = long long;
#define ls (id->lchild)
#define rs (id->rchild)
template<class Info, class Tag, class T = i64>
struct SegmentTree{
    struct Node {
        Node* lchild = nullptr;
        Node* rchild = nullptr;
        Info info;
        Tag tag;
    };
    T L, R;
    Node* root = nullptr;

    SegmentTree(T l, T r) : L(l), R(r) {
        root = new Node();
        (root->info).len = r - l + 1;
    }
    SegmentTree(T n) : SegmentTree(0, n) {}

    void apply(Node* id,const Tag &dx) {
        id->info.apply(dx);
        id->tag.apply(dx);
    }

    void pushdown(Node* id, T l, T r) {
        T mid = l + r >> 1;
        if (ls == nullptr) {
```

```
            ls = new Node();
            ls->info.len = mid - l + 1;
        }
        if (rs == nullptr) {
            rs = new Node();
            rs->info.len = r - mid;
        }
        apply(ls, id->tag);
        apply(rs, id->tag);
        id->tag = Tag();
    }

    void pushup(Node* id) {
        id->info = ls->info + rs->info;
    }

    void rangeUpdate(Node* id, T l, T r, T x, T y,const Tag &dx) {

        if (x <= l && y >= r) {
            apply(id, dx);
            return;
        }

        T mid = l + r >> 1;

        pushdown(id, l, r);

        if (x <= mid) {
            rangeUpdate(ls, l, mid, x, y, dx);
        }
        if (y > mid) {
            rangeUpdate(rs, mid + 1, r, x, y, dx);
        }
        pushup(id);
    }
    void modify(Node* id, T l, T r, T pos, const Info &val) {
        if (l == r) {
             id->info = val;
             return;
        }
        pushdown(id, l, r);
        T mid = l + r >> 1;
        if (pos <= mid) {
            modify(ls, l, mid, pos, val);
        } else {
            modify(rs, mid + 1, r, pos, val);
        }
        pushup(id);
    }


    Info rangeQuery(Node* id, T l, T r, T x, T y) {
        if (l >= x && r <= y) {
            return id->info;
        }
        Info res;
        T mid = l + r >> 1;

        pushdown(id, l, r);

        if (x <= mid) {
            res = res + rangeQuery(ls, l, mid, x, y);
        }
        if (y > mid) {
            res = res + rangeQuery(rs, mid + 1, r, x, y);
        }
        return res;
    }


    void rangeUpdate(T x, T y, const Tag &dx) {
        rangeUpdate(root, L, R, x, y, dx);
    }
    void modify(T pos, const Info &val) {
        modify(root, L, R, pos, val);
    }
    Info rangeQuery(T x, T y) {
        return rangeQuery(root, L, R, x, y);
    }
};
#undef ls
#undef rs
struct Tag {
    i64 add = 0;
    void apply(const Tag &dx) {
        add += dx.add;
    }
};

struct Info{
    i64 sum = 0;
    i64 len = 0;

    void apply(const Tag &dx) {
        sum += (dx.add * len);
```

```
    }
};
Info operator+(const Info a, const Info b) {
    Info res;
    res.sum = a.sum + b.sum;
    res.len = a.len + b.len;
    return res;
}



void solve() {
    int n, m;
    cin >> n >> m;
    SegmentTree<Info, Tag> seg(1, n);
    while(m--) {
        int op;
        cin >> op;
        if (op == 1) {
            int l, r, k;
            cin >> l >> r >> k;
            Tag tag;
            tag.add = k;
            seg.rangeUpdate(l, r, tag);
            // cout << 1 << endl;
        } else {
            int l, r;
            cin >> l >> r;
            cout << seg.rangeQuery(l, r).sum << '\n';
            // cout << 2 << endl;
        }
    }







}


int main() {
    std::ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);

    int T = 1;
    //cin >> T;
    while (T--) solve();

    return 0;
}
```

1.6 区间修改线段树

```
//线段树，区间修改，区间查询
//https://www.luogu.com.cn/problem/P3372
template<typename Info, typename Tag>
struct SegmentTree {
#define ls (id<<1)
#define rs (id<<1|1)
    SegmentTree() = default;
    SegmentTree(int n) : n(n), info(n << 2), tag(n << 2), len(n << 2) {}
    SegmentTree(const SegmentTree<Info, Tag> &o) : n(o.n), info(o.info), tag(o.tag)
    {}
    template<typename T>
    SegmentTree(const std::vector<T> &init) : SegmentTree((int)init.size()) {
        auto build = [&](auto self, int id, int l, int r) ->void {
            len[id] = r - l + 1;
            if(l == r) {
                info[id] = init[l];
                return;
            }
            int mid = (l + r) / 2;
            self(self, ls, l, mid);
            self(self, rs, mid + 1, r);
            pushup(id);
        };
        build(build, 1, 0, n - 1);
    }
    void apply(int id, const Tag &dx) {
        info[id].apply(dx, len[id]);
        tag[id].apply(dx);
    }
    void pushup(int id) {
        info[id] = info[ls] + info[rs];
    }
    void pushdown(int id) {
        apply(ls, tag[id]);
        apply(rs, tag[id]);
        tag[id] = Tag();
    }

    void modify(int id, int l, int r, int pos, const Info &val) {
        if(l == r) {
            info[id] = val;
            return;
        }
        pushdown(id);
```

```
        int mid = (l + r) / 2;
        if(pos <= mid) {
            modify(ls, l, mid, pos, val);
        } else {
            modify(rs, mid + 1, r, pos, val);
        }
        pushup(id);
    }

    void rangeUpdate(int id, int l, int r, int x, int y, const Tag &dx) {
        if(x <= l && r <= y) {
            apply(id, dx);
            return;
        }
        int mid = (l + r) / 2;
        pushdown(id);
        if(x <= mid) {
            rangeUpdate(ls, l, mid, x, y, dx);
        }
        if(y > mid) {
            rangeUpdate(rs, mid + 1, r, x, y, dx);
        }
        pushup(id);
    }
    Info rangeQuery(int id, int l, int r, int x, int y) {
        if(x <= l && r <= y) {
            return info[id];
        }
        int mid = (l + r) / 2;
        pushdown(id);
        Info res;
        if(x <= mid) {
            res = res + rangeQuery(ls, l, mid, x, y);
        }
        if(y > mid) {
            res = res + rangeQuery(rs, mid + 1, r, x, y);
        }
        return res;
    }


    void rangeUpdate(int l, int r, const Tag &dx) {
        rangeUpdate(1, 0, n - 1, l, r, dx);
    }
    void update(int pos, const Tag &dx) {
        rangeUpdate(pos, pos, dx);
    }
    Info rangeQuery(int l, int r) {
        return rangeQuery(1, 0, n - 1, l, r);
    }
    Info query(int pos) {
        return rangeQuery(pos, pos);
    }

    void modify(int pos, const Info &val) {
        return modify(1, 0, n - 1, pos, val);
    }
#undef ls
#undef rs
    int n;
    std::vector<Info> info;
    std::vector<Tag> tag;
    std::vector<int> len;
};

constexpr i64 INF = 4E18;
i64 mod = 1e9 + 7;
struct Tag {
    i64 add = 0;
    i64 mul = 1;
    void apply(const Tag &dx) {
        mul = (mul * dx.mul) % mod;
        add = (add * dx.mul + dx.add) % mod;
    }
};

struct Info {
    i64 sum = 0;
    Info() {}
    Info(i64 x) : sum(x) {};
    void apply(const Tag &dx, const int &len) {
        sum = (sum * dx.mul + dx.add * len) % mod;
    }
};

Info operator+(const Info &x, const Info &y) {
    Info res;
    res.sum = (x.sum + y.sum) % mod;
    return res;
}
```

1.7 波纹疾走树

```
#include<bits/stdc++.h>
using namespace std;
using i64 = long long;
```

```cpp
struct BitRank {
    // block 管理一行一行的bit
    std::vector<unsigned long long> block;
    std::vector<unsigned int> count;
    BitRank() {}
    // 位向量长度
    void resize(const unsigned int num) {
        block.resize(((num + 1) >> 6) + 1, 0);
        count.resize(block.size(), 0);
    }
    // 设置i位bit
    void set(const unsigned int i, const unsigned long long val) {
        block[i >> 6] |= (val << (i & 63));
    }
    void build() {
        for (unsigned int i = 1; i < block.size(); i++) {
            count[i] = count[i - 1] + __builtin_popcountll(block[i - 1]);
        }
    }
    // [0, i) 1的个数
    unsigned int rank1(const unsigned int i) const {
        return count[i >> 6] + __builtin_popcountll(block[i >> 6] & ((1ULL << (i & 63)) - 1ULL));
    }
    // [i, j) 1的个数
    unsigned int rank1(const unsigned int i, const unsigned int j) const {
        return rank1(j) - rank1(i);
    }
    // [0, i) 0的个数
    unsigned int rank0(const unsigned int i) const {
        return i - rank1(i);
    }
    // [i, j) 0的个数
    unsigned int rank0(const unsigned int i, const unsigned int j) const {
        return rank0(j) - rank0(i);
    }
};


class WaveletMatrix {
private:
    unsigned int height;
    std::vector<BitRank> B;
    std::vector<int> pos;
public:
    WaveletMatrix() {}
    WaveletMatrix(std::vector<int> vec) : WaveletMatrix(vec, *std::max_element(vec.begin(), vec.end()) + 1) {}
    // sigma: 字母表大小(字符串的话), 数字序列的话是数的种类
    WaveletMatrix(std::vector<int> vec, const unsigned int sigma) {
        height = (sigma == 1) ? 1 : (64 - __builtin_clzll(sigma - 1));
        B.resize(height), pos.resize(height);
        for (unsigned int i = 0; i < height; ++i) {
            B[i].resize(vec.size());
            for (unsigned int j = 0; j < vec.size(); ++j) {
                B[i].set(j, get(vec[j], height - i - 1));
            }
            B[i].build();
            auto it = stable_partition(vec.begin(), vec.end(), [&](int c) {
                return !get(c, height - i - 1);
            });
            pos[i] = it - vec.begin();
        }
    }

    int get(const int val, const int i) {
        return (val >> i) & 1;
    }

    // [l, r] 中val出现的频率
    int rank(const int l, const int r, const int val) {
        return rank(r, val) - rank(l - 1, val);
    }

    // [0, i] 中val出现的频率
    int rank(int i, int val) {
        ++i;
        int p = 0;
        for (unsigned int j = 0; j < height; ++j) {
            if (get(val, height - j - 1)) {
                p = pos[j] + B[j].rank1(p);
                i = pos[j] + B[j].rank1(i);
            } else {
                p = B[j].rank0(p);
                i = B[j].rank0(i);
            }
        }
      return i - p;
    }

    // [l, r] 中第k小, 只算非负数
    int kth(int l, int r, int k) {
        ++r;
        int res = 0;
        for (unsigned int i = 0; i < height; ++i) {
            const int j = B[i].rank0(l, r);
```

```
            if (j >= k) {
                l = B[i].rank0(l);
                r = B[i].rank0(r);
            } else {
                l = pos[i] + B[i].rank1(l);
                r = pos[i] + B[i].rank1(r);
                k -= j;
                res |= (1 << (height - i - 1));
            }
        }
        return res;
    }

    // [l,r] 在[a, b] 值域的数字个数 数组中只可有非负数
    int rangeFreq(const int l, const int r, const int a, const int b) {
        return rangeFreq(l, r + 1, a, b + 1, 0, 1 << height, 0);
    }
    int rangeFreq(const int i, const int j, const int a, const int b, const int l, const int r, const int x) {
        if (i == j || r <= a || b <= l) return 0;
        const int mid = (l + r) >> 1;
        if (a <= l && r <= b) {
            return j - i;
        } else {
            const int left = rangeFreq(B[x].rank0(i), B[x].rank0(j), a, b, l, mid, x + 1);
            const int right = rangeFreq(pos[x] + B[x].rank1(i), pos[x] + B[x].rank1(j), a, b, mid, r, x + 1);
            return left + right;
        }
    }

    // [l,r] 在[a,b] 值域内存在的最小值是什么，不存在返回-1，只支持非负整数
    int rangeMin(int l, int r, int a, int b) {
        return rangeMin(l, r + 1, a, b + 1, 0, 1 << height, 0, 0);
    }
    int rangeMin(const int i, const int j, const int a, const int b, const int l, const int r, const int x, const int val) {
        if (i == j || r <= a || b <= l) return -1;
        if (r - l == 1) return val;
        const int mid = (l + r) >> 1;
        const int res = rangeMin(B[x].rank0(i), B[x].rank0(j), a, b, l, mid, x + 1, val);
        if (res < 0) {
            return rangeMin(pos[x] + B[x].rank1(i), pos[x] + B[x].rank1(j), a, b, mid, r, x + 1, val + (1 << (height - x - 1)));
        } else {
            return res;
        }
    }
};

void solve() {
    int n, m;
    cin >> n >> m;
    vector<int> a(n + 1);
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
    }
    WaveletMatrix tree(a);
    while(m--) {
        int l, r, k;
        cin >> l >> r >> k;
        cout << tree.kth(l, r, k) << '\n';
    }



}


int main() {
    std::ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);

    int T = 1;
    //cin >> T;
    while (T--) solve();

    return 0;
}
```

1.8 笛卡尔树

```
#include<bits/stdc++.h>
using namespace std;
using i64 = long long;
template<typename T>
struct Cartesian {
    Cartesian() = default;
    Cartesian(const std::vector<T> &v)
        : ls(v.size(), -1), rs(v.size(), -1) {
        std::stack<int> stk;
        for(int i = 1; i < v.size(); ++i) { // 维护下标 1~n
            while(!stk.empty() && v[i] < v[stk.top()]) {
                stk.pop();
```

```
            }
            if(stk.empty()) {
                ls[i] = root;
                root = i;
            } else {
                ls[i] = rs[stk.top()];
                rs[stk.top()] = i;
            }
            stk.push(i);
        }
    }
    int root = -1;
    std::vector<int> ls, rs;
};

void solve() {
    int n;
    cin >> n;
    vector<int> a(n + 1);
    for (int i = 1; i <= n; i++) cin >> a[i];
    Cartesian tree(a);
    auto &ls = tree.ls, &rs = tree.rs;
    i64 ansl = 0, ansr = 0;
    for (int i = 1; i <= n; i++) {
        // cout << ls[i] << endl;
        ansl ^= 1ll * i * ((ls[i] == -1 ? 0 : ls[i]) + 1);
        ansr ^= 1ll * i * ((rs[i] == -1 ? 0 : rs[i]) + 1);
    }

    cout << ansl << ' ' << ansr;









}


int main() {
    std::ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);

    int T = 1;
    //cin >> T;
    while (T--) solve();

    return 0;
}
```

1.9 线段树 (未修改) (SegmentTree+Info 初始赋值 + 单点修改 + 查找前驱后继)

```
template<class Info>
struct SegmentTree {
    int n;
    std::vector<Info> info;
    SegmentTree() : n(0) {}
    SegmentTree(int n_, Info v_ = Info()) {
        init(n_, v_);
    }
    template<class T>
    SegmentTree(std::vector<T> init_) {
        init(init_);
    }
    void init(int n_, Info v_ = Info()) {
        init(std::vector(n_, v_));
    }
    template<class T>
    void init(std::vector<T> init_) {
        n = init_.size();
        info.assign(4 << std::__lg(n), Info());
        std::function<void(int, int, int)> build = [&](int p, int l, int r) {
            if (r - l == 1) {
                info[p] = init_[l];
                return;
            }
            int m = (l + r) / 2;
            build(2 * p, l, m);
            build(2 * p + 1, m, r);
            pull(p);
        };
        build(1, 0, n);
    }
    void pull(int p) {
        info[p] = info[2 * p] + info[2 * p + 1];
    }
    void modify(int p, int l, int r, int x, const Info &v) {
        if (r - l == 1) {
            info[p] = v;
            return;
        }
        int m = (l + r) / 2;
        if (x < m) {
            modify(2 * p, l, m, x, v);
        } else {
            modify(2 * p + 1, m, r, x, v);
        }
```

```
        pull(p);
    }
    void modify(int p, const Info &v) {
        modify(1, 0, n, p, v);
    }
    Info rangeQuery(int p, int l, int r, int x, int y) {
        if (l >= y || r <= x) {
            return Info();
        }
        if (l >= x && r <= y) {
            return info[p];
        }
        int m = (l + r) / 2;
        return rangeQuery(2 * p, l, m, x, y) + rangeQuery(2 * p + 1, m, r, x, y);
    }
    Info rangeQuery(int l, int r) {
        return rangeQuery(1, 0, n, l, r);
    }
    template<class F>
    int findFirst(int p, int l, int r, int x, int y, F &&pred) {
        if (l >= y || r <= x) {
            return -1;
        }
        if (l >= x && r <= y && !pred(info[p])) {
            return -1;
        }
        if (r - l == 1) {
            return l;
        }
        int m = (l + r) / 2;
        int res = findFirst(2 * p, l, m, x, y, pred);
        if (res == -1) {
            res = findFirst(2 * p + 1, m, r, x, y, pred);
        }
        return res;
    }
    template<class F>
    int findFirst(int l, int r, F &&pred) {
        return findFirst(1, 0, n, l, r, pred);
    }
    template<class F>
    int findLast(int p, int l, int r, int x, int y, F &&pred) {
        if (l >= y || r <= x) {
            return -1;
        }
        if (l >= x && r <= y && !pred(info[p])) {
            return -1;
        }
        if (r - l == 1) {
            return l;
        }
        int m = (l + r) / 2;
        int res = findLast(2 * p + 1, m, r, x, y, pred);
        if (res == -1) {
            res = findLast(2 * p, l, m, x, y, pred);
        }
        return res;
    }
    template<class F>
    int findLast(int l, int r, F &&pred) {
        return findLast(1, 0, n, l, r, pred);
    }
};

constexpr int inf = 1E9 + 1;
struct Info {
    int max = -inf;
    int min = inf;
};
Info operator+(const Info &a, const Info &b) {
    return { std::max(a.max, b.max), std::min(a.min, b.min) };
}
```

1.10 线段树（SegmentTree+Info 初始赋值 + 单点修改 + 查找前驱后继）

```
template<class Info>
struct SegmentTree {
    int n;
    std::vector<Info> info;
    SegmentTree() : n(0) {}
    SegmentTree(int n_, Info v_ = Info()) {
        init(n_, v_);
    }
    template<class T>
    SegmentTree(std::vector<T> init_) {
        init(init_);
    }
    void init(int n_, Info v_ = Info()) {
        init(std::vector(n_, v_));
    }
    template<class T>
    void init(std::vector<T> init_) {
        n = init_.size();
        info.assign(4 << std::__lg(n), Info());
```

```
        std::function<void(int, int, int)> build = [&](int p, int l, int r) {
            if (r - l == 1) {
                info[p] = init_[l];
                return;
            }
            int m = (l + r) / 2;
            build(2 * p, l, m);
            build(2 * p + 1, m, r);
            pull(p);
        };
        build(1, 0, n);
    }
    void pull(int p) {
        info[p] = info[2 * p] + info[2 * p + 1];
    }
    void modify(int p, int l, int r, int x, const Info &v) {
        if (r - l == 1) {
            info[p] = v;
            return;
        }
        int m = (l + r) / 2;
        if (x < m) {
            modify(2 * p, l, m, x, v);
        } else {
            modify(2 * p + 1, m, r, x, v);
        }
        pull(p);
    }
    void modify(int p, const Info &v) {
        modify(1, 0, n, p, v);
    }
    Info rangeQuery(int p, int l, int r, int x, int y) {
        if (l >= y || r <= x) {
            return Info();
        }
        if (l >= x && r <= y) {
            return info[p];
        }
        int m = (l + r) / 2;
        return rangeQuery(2 * p, l, m, x, y) + rangeQuery(2 * p + 1, m, r, x, y);
    }
    Info rangeQuery(int l, int r) {
        return rangeQuery(1, 0, n, l, r + 1);
    }
    template<class F>
    int findFirst(int p, int l, int r, int x, int y, F &&pred) {
        if (l >= y || r <= x) {
            return -1;
        }
        if (l >= x && r <= y && !pred(info[p])) {
            return -1;
        }
        if (r - l == 1) {
            return l;
        }
        int m = (l + r) / 2;
        int res = findFirst(2 * p, l, m, x, y, pred);
        if (res == -1) {
            res = findFirst(2 * p + 1, m, r, x, y, pred);
        }
        return res;
    }
    template<class F>
    int findFirst(int l, int r, F &&pred) {
        return findFirst(1, 1, n, l, r + 1, pred);
    }
    template<class F>
    int findLast(int p, int l, int r, int x, int y, F &&pred) {
        if (l >= y || r <= x) {
            return -1;
        }
        if (l >= x && r <= y && !pred(info[p])) {
            return -1;
        }
        if (r - l == 1) {
            return l;
        }
        int m = (l + r) / 2;
        int res = findLast(2 * p + 1, m, r, x, y, pred);
        if (res == -1) {
            res = findLast(2 * p, l, m, x, y, pred);
        }
        return res;
    }
    template<class F>
    int findLast(int l, int r, F &&pred) {
        return findLast(1, 0, n, l, r + 1, pred);
    }
};

constexpr int inf = 1E9 + 1;
struct Info {
    int max = -inf;
    int min = inf;
};
Info operator+(const Info &a, const Info &b) {
```

```
    return { std::max(a.max, b.max), std::min(a.min, b.min) };
}
```

2 树上问题

2.1 LCA(倍增)

```
vector<int> dep(tot + 1), vis1(tot + 1);
vector<array<int, 31>> anc(n + 1);

auto dfs1 = [&] (auto self, int u, int p) -> void{
    vis1[u] = 1;
    anc[u][0] = p;
    dep[u] = dep[p] + 1;


    for(int i = 1; i <= 30; i++) {
        anc[u][i] = anc[anc[u][i - 1]][i - 1];
    }

    for(auto v : edge[u]) {
        if(v == p) continue;
        self(self, v, u);
    }
};
//森林
for(int i = 1; i <= tot; i++) {
    if(!vis1[i]) dfs1(dfs1, i, 0);
}

auto lca = [&] (int u, int v) {
    int zu = u, zv = v;
    if(dep[zu] < dep[zv]) swap(zu, zv);

    int gap = dep[zu] - dep[zv];
    for(int i = 0; i <= 30; i++) {
        if((gap >> i) & 1) zu = anc[zu][i];
    }
    if(zu == zv) return zu;

    for(int i = 30; i >= 0; i--) {
        if(anc[zu][i] != anc[zv][i]) {
            zu = anc[zu][i];
            zv = anc[zv][i];
        }
    }
    return anc[zu][0];
};
```

2.2 LCA

```
// 倍增求LCA
// 预处理O(nlogn) 单次查询O(logn)
void solve1(){
    int n, m, s;
    cin >> n >> m >> s;
    vector<vector<int>> edge(n + 1);
    for(int i = 1; i < n; i++) {
        int u, v;
        cin >> u >> v;
        edge[u].push_back(v);
        edge[v].push_back(u);
    }

    vector<int> dep(n + 1);
    vector<vector<int>> anc(n + 1, vector<int>(31));
    auto dfs = [&] (auto self, int x, int f) -> void {
        dep[x] = dep[f] + 1;
        anc[x][0] = f;
        for(int i = 1; i < 31; i++) {
            anc[x][i] = anc[anc[x][i - 1]][i - 1];
        }

        for(auto y : edge[x]) {
            if(y == f) continue;
            self(self, y, x);
        }

    };
    dfs(dfs, s, 0);
    auto lca = [&] (int x, int y) -> int{
        if(dep[x] > dep[y]) swap(x, y);
        int gap = dep[y] - dep[x];
        for(int i = 0; i <= 30; i++) {
            if((gap >> i) & 1) y = anc[y][i];
        }
        if(x == y) return x;
        for(int i = 30; i >= 0; i--) {
            if(anc[x][i] != anc[y][i]) {
                x = anc[x][i];
                y = anc[y][i];
            }
        }
        return anc[x][0];

```

```
    };
    while(m--) {
        int x, y;
        cin >> x >> y;
        cout << lca(x, y) << '\n';
    }



}




//tarjan离线求LCA
//节点数n 查询数m O(n + m)
void solve2() {
    int n, m, s;
    cin >> n >> m >> s;
    vector<vector<int>> edge(n + 1);
    for(int i = 1; i < n; i++) {
        int u, v;
        cin >> u >> v;
        edge[u].push_back(v);
        edge[v].push_back(u);
    }
    vector<int> p(n + 1);
    for(int i = 1; i <= n; i++) p[i] = i;
    auto chazu = [&](auto self, int x) -> int {
        if(x != p[x]) p[x] = self(self, p[x]);
        return p[x];
    };
    vector<int> ans(m + 1), st(n + 1);
    vector<vector<info>> que(n + 1);
    for(int i = 1; i <= m; i++) {
        int a, b;
        cin >> a >> b;
        que[a].push_back({b, i});
        que[b].push_back({a, i});
    }

    auto dfs = [&] (auto self, int x, int f) -> void {
        st[x] = 1;

        for(auto y : edge[x]) {
            if(y == f) continue;
            self(self, y, x);
        }
        for(auto [y, idx] : que[x]) {
            if(st[y]) ans[idx] = chazu(chazu, y);
        }


        p[x] = f;


    };
    dfs(dfs, s, 0);

    for(int i = 1; i <= m; i++) cout << ans[i] << '\n';




}
```

2.3 树上启发式合并

```
#include<bits/stdc++.h>
using namespace std;
using i64 = long long;
// 不能使用0下标
// 传n + 1 使用下标 1~n
struct HLD {
    HLD(const int &n) : n(n), edge(n) {
        fa = dep = son = sz = top = dfn = out = rnk = std::vector<int>(n);
    }

    void addEdge(const int &x, const int &y) {
        edge[x].push_back(y);
        edge[y].push_back(x);
    }

    void dfs1(int u, int p) {
        sz[u] = 1;
        fa[u] = p;
        dep[u] = dep[p] + 1;
        for(const auto &v : edge[u]) {
            if(v == fa[u]) continue;
            dfs1(v, u);
            sz[u] += sz[v];
            if(sz[son[u]] < sz[v]) {
                son[u] = v;
            }
        }
    }
```

```
    void dfs2(int u, int t, int &dfncnt) {
        dfncnt++;
        dfn[u] = dfncnt;
        rnk[dfncnt] = u;
        top[u] = t;
        if(son[u] == 0) return;
        dfs2(son[u], t, dfncnt);
        for(const auto &v : edge[u]) {
            if(v == fa[u] || v == son[u]) continue;
            dfs2(v, v, dfncnt);
        }
    }
    void work(int root = 1) {
        int dfncnt = 0;
        dfs1(root, 0);
        dfs2(root, root, dfncnt);
    }
    int lca(int u, int v) {
        while(top[u] != top[v]) {
            if(dep[top[u]] < dep[top[v]]) {
                std::swap(u, v);
            }
            u = fa[top[u]];
        }
        return (dep[u] < dep[v] ? u : v);
    }

    int dis(int x, int y) {
        return dep[x] + dep[y] - 2 * dep[lca(x, y)];
    }

    // int kth(int id, int k) {
    //  if(k > dep[id]) return 0;
    //  while(dep[id] - dep[top[id]] + 1 <= k) {
    //      k -= (dep[id] - dep[top[id]] + 1);
    //      id = fa[top[id]];
    //  }
    //  return rnk[dfn[id] - k];
    // }

    vector<vector<int>> edge;
    vector<int> fa, dep, son, sz, top, dfn, out, rnk;
    int n;
};

void solve() {
    int n;
    cin >> n;
    HLD tree(n + 1);
    vector<vector<int>> edge(n + 1);
    for (int i = 1; i < n; i++) {
        int u, v;
        cin >> u >> v;
        edge[u].push_back(v);
        edge[v].push_back(u);
        tree.addEdge(u, v);
    }
    vector<int> a(n + 1);
    for (int i = 1; i <= n; i++) cin >> a[i];
    vector<int> cnt(n + 1);
    int dif = 0;
    vector<int> ans(n + 1);
    tree.work(1);
    auto &dfn = tree.dfn;
    auto &rnk = tree.rnk;
    auto &son = tree.son;
    auto &sz = tree.sz;

    auto dfs = [&] (auto self, int u, int p, int keep) -> void {

        for (auto v : edge[u]) {
            if (v == p || v == son[u]) continue;
            self(self, v, u, 0);
        }
        if (son[u]) self(self, son[u], u, 1);
        cnt[a[u]]++;
        if (cnt[a[u]] == 1) dif++;

        for (auto v : edge[u]) {
            if (v == p || v == son[u]) continue;
            for (int i = dfn[v]; i <= dfn[v] + sz[v] - 1; i++) {
                cnt[a[rnk[i]]]++;
                if (cnt[a[rnk[i]]] == 1) dif++;
            }
        }
        ans[u] = dif;

        if (!keep) {
            dif = 0;
            for (int i = dfn[u]; i <= dfn[u] + sz[u] - 1; i++) {
                cnt[a[rnk[i]]]--;
            }
        }


    };
```

```

    dfs(dfs, 1, 0, 1);
    int m;
    cin >> m;
    while(m--) {
        int x;
        cin >> x;
        cout << ans[x] << '\n';
    }




}


int main() {
    std::ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);

    int T = 1;
    //cin >> T;
    while (T--) solve();

    return 0;
}
```

2.4 重链剖分

```
// 不能使用0下标
// 传n + 1 使用下标 1~n
// son如果没有重儿子则为0
struct HLD {
    HLD(const int &n) : n(n), edge(n) {
        fa = dep = son = sz = top = dfn = out = rnk = std::vector<int>(n);
    }

    void addEdge(const int &x, const int &y) {
        edge[x].push_back(y);
        edge[y].push_back(x);
    }

    void dfs1(int u, int p) {
        sz[u] = 1;
        fa[u] = p;
        dep[u] = dep[p] + 1;
        for(const auto &v : edge[u]) {
            if(v == fa[u]) continue;
            dfs1(v, u);
            sz[u] += sz[v];
            if(sz[son[u]] < sz[v]) {
                son[u] = v;
            }
        }
    }
    void dfs2(int u, int t, int &dfncnt) {
        dfncnt++;
        dfn[u] = dfncnt;
        rnk[dfncnt] = u;
        top[u] = t;
        if(son[u] == 0) return;
        dfs2(son[u], t, dfncnt);
        for(const auto &v : edge[u]) {
            if(v == fa[u] || v == son[u]) continue;
            dfs2(v, v, dfncnt);
        }
    }
    void work(int root = 1) {
        int dfncnt = 0;
        dfs1(root, 0);
        dfs2(root, root, dfncnt);
    }
    int lca(int u, int v) {
        while(top[u] != top[v]) {
            if(dep[top[u]] < dep[top[v]]) {
                std::swap(u, v);
            }
            u = fa[top[u]];
        }
        return (dep[u] < dep[v] ? u : v);
    }

    int dis(int x, int y) {
        return dep[x] + dep[y] - 2 * dep[lca(x, y)];
    }

    // int kth(int id, int k) {
    //  if(k > dep[id]) return 0;
    //  while(dep[id] - dep[top[id]] + 1 <= k) {
    //      k -= (dep[id] - dep[top[id]] + 1);
    //      id = fa[top[id]];
    //  }
    //  return rnk[dfn[id] - k];
    // }

    vector<vector<int>> edge;
    vector<int> fa, dep, son, sz, top, dfn, out, rnk;
```

```
    int n;
};
```

3 图论

3.1 bellmanfloyd

```
#include <bits/stdc++.h>
using namespace std;
const int M = 1e4+1;
const int N = 550;
int n,m,k;
struct edge{
    int a,b,w;
}edges[M];
int dist[N];
int backup[N];
int bf(){

    memset(dist,0x3f,sizeof(dist));
    dist[1] = 0;
    for(int i = 1;i<=k;i++){
        memcpy(backup,dist,sizeof(dist));
        for(int j = 1;j<=m;j++){
            int a = edges[j].a;
            int b = edges[j].b;
            int w = edges[j].w;
            dist[b] = min(dist[b],backup[a]+w);
        }
    }
    if (dist[n] > 0x3f3f3f3f / 2) return -0x3f3f3f3f;
    else return dist[n];
}
int main()
{
    cin>>n>>m>>k;

    for(int i = 1;i<=m;i++){
        int a,b,w;
        cin>>a>>b>>w;
        edges[i] = {a,b,w};
    }

    int t = bf();

    if(t==-0x3f3f3f3f) cout<<"impossible"<<endl;
    else cout<<t<<endl;
    return 0;
}
```

3.2 dijk

```
#include <bits/stdc++.h>
using namespace std;
using pii = pair<int,int>;
const int N = 2e5+10;
bool st[N];
int main()
{
    int n,m;
    cin>>n>>m;
    int dist[N];
    memset(dist,0x3f,sizeof(dist));
    vector<vector<pii>> gra(n+1);
    while(m--){
        int a,b,c;
        cin>>a>>b>>c;
        gra[a].push_back({b,c});
    }
    priority_queue<pii,vector<pii>,greater<pii>> q;
    dist[1] = 0;
    q.push({0,1});
    while(!q.empty()){
        pii tnod = q.top();
        q.pop();
        int nod = tnod.second;
        int d = tnod.first;
        if(st[nod]) continue;
        st[nod] = true;
        for(auto [next_nod,vlen] : gra[nod]){
            if(dist[next_nod] > dist[nod]+vlen){
                dist[next_nod] = dist[nod]+vlen;
                q.push({dist[next_nod],next_nod});
            }
        }
    }
    if(dist[n]==0x3f3f3f3f) cout<<"-1"<<endl;
    else cout<<dist[n]<<endl;
    return 0;
}
```

3.3 dinic 求最大流

```
// 复杂度上界 O((n ^ 2) * m)
```

```cpp
// 求最小割容量即为最大流
// 最小割的最小化割边数量：先跑一遍dinic，把没有流满的边改为dinic，流满的边改为
// 1，重新跑一边dinic即为最小割边数量
// 没有最小割限制，直接把所有边容量设为1，跑一边dinic
template<class T>
struct MaxFlow {
    struct _Edge {
        int to;
        T cap;
        _Edge(int to, T cap) : to(to), cap(cap) {}
    };

    int n;
    std::vector<_Edge> e;
    std::vector<std::vector<int>> g; //g存每个点连的边的e数组下标
    std::vector<int> cur, h; //h是点的深度

    MaxFlow() {}
    MaxFlow(int n) {
        init(n);
    }

    void init(int n) {
        this->n = n;
        e.clear();
        g.assign(n, {});
        cur.resize(n);
        h.resize(n);
    }

    bool bfs(int s, int t) {
        h.assign(n, -1);
        std::queue<int> que;
        h[s] = 0;
        que.push(s);
        while (!que.empty()) {
            const int u = que.front();
            que.pop();
            for (int i : g[u]) {
                auto [v, c] = e[i];
                if (c > 0 && h[v] == -1) {
                    h[v] = h[u] + 1;
                    if (v == t) {
                        return true;
                    }
                    que.push(v);
                }
            }
        }
        return false;
    }

    T dfs(int u, int t, T f) {
        if (u == t) {
            return f;
        }
        auto r = f;
        for (int &i = cur[u]; i < int(g[u].size()); ++i) {
            const int j = g[u][i];
            auto [v, c] = e[j];
            if (c > 0 && h[v] == h[u] + 1) {
                auto a = dfs(v, t, std::min(r, c));
                e[j].cap -= a;
                e[j ^ 1].cap += a;
                r -= a;
                if (r == 0) {
                    return f;
                }
            }
        }
        return f - r;
    }
    void addEdge(int u, int v, T c) {
        g[u].push_back(e.size());
        e.emplace_back(v, c);
        g[v].push_back(e.size());
        e.emplace_back(u, 0);
    }
    T flow(int s, int t) { //求最大流
        T ans = 0;
        while (bfs(s, t)) {
            cur.assign(n, 0);
            ans += dfs(s, t, std::numeric_limits<T>::max());
        }
        return ans;
    }

    std::vector<bool> minCut() { //求一个最小割的划分， c[i] = 1在S集合，= 0在T集合
        std::vector<bool> c(n);
        for (int i = 0; i < n; i++) {
            c[i] = (h[i] != -1);
        }
        return c;
    }

    struct Edge {
```

```
        int from;
        int to;
        T cap;
        T flow;
    };
    std::vector<Edge> edges() {
        std::vector<Edge> a;
        for (int i = 0; i < e.size(); i += 2) {
            Edge x;
            x.from = e[i + 1].to;
            x.to = e[i].to;
            x.cap = e[i].cap + e[i + 1].cap;
            x.flow = e[i + 1].cap;
            a.push_back(x);
        }
        return a;
    }
};
```

3.4 kruskal

```
#include <iostream>
#include <algorithm>
using namespace std;
const int N = 1e5+5;
const int M = N*2;
int n,m;
int p[N];
struct edge{
    int a,b,w;
    bool operator < (const edge &W) const {
        return w<W.w;
    }
}edges[M];
int find(int x){
   if(p[x]!=x) p[x] = find(p[x]);
   return p[x];
}
int main()
{
    cin>>n>>m;
    for(int i = 1;i<=m;i++){
        int a,b,c;
        cin>>a>>b>>c;
        edges[i] = {a,b,c};
    }
    sort(edges+1,edges+1+m);
    int res = 0;
    int cnt = 0;
    for(int i = 1;i<=n;i++) p[i] = i;
    for(int i = 1;i<=m;i++){
        int a = edges[i].a;
        int b = edges[i].b;
        int c = edges[i].w;
        a = find(a);
        b = find(b);
        if(a==b){
            ;
        }else{
            p[a] = p[b];
            cnt++;
            res+=c;
        }
    }
    if(cnt!=n-1) cout<<"impossible";
    else cout<<res<<endl;
    return 0;
}
```

3.5 prim

```
#include <iostream>
#include <cstring>
using namespace std;
const int N = 510;
const int M = 1e5+10;
int mp[N][N];
int dist[N];
bool st[N];
int n,m;

int prim(){
      memset(dist, 0x3f, sizeof dist);

    int res = 0;
    for (int i = 0; i < n; i ++ )
    {
        int t = -1;
        for (int j = 1; j <= n; j ++ )
            if (!st[j] && (t == -1 || dist[t] > dist[j]))
                t = j;

        if (i && dist[t] == 0x3f3f3f3f) return 0x3f3f3f3f;

        if (i) res += dist[t];
        st[t] = true;
```

```

        for (int j = 1; j <= n; j ++ ) dist[j] = min(dist[j], mp[t][j]);
    }

    return res;
}
int main()
{
    cin>>n>>m;
    memset(mp,0x3f,sizeof(mp));
    memset(dist,0x3f,sizeof(dist));
    for(int i = 1;i<=m;i++){
        int u,v,w;
        cin>>u>>v>>w;
        mp[u][v] = min(mp[u][v],w);
        mp[v][u] = min(mp[v][u],w);
    }
    int t = prim();
    if(t==0x3f3f3f3f) cout<<"impossible"<<endl;
    else cout<<t<<endl;
    return 0;
}
```

3.6 tarjan 割点

```
#include<bits/stdc++.h>
using namespace std;
using i64 = long long;


struct VBCC {
    int n;
    vector<vector<int>> adj;
    vector<int> stk;
    vector<vector<int>> ecc;
    vector<int> dfn, low;
    int cur, tot;
    VBCC(int n) {
        init(n);
    }

    void init(int n){
        this->n = n;
        adj.assign(n + 1, {});
        ecc.assign(n + 1, {});
        dfn.assign(n + 1, 0);
        low.resize(n + 1);
        stk.clear();
        cur = tot = 0;
    }

    void addEdge(int u, int v) {
        adj[u].push_back(v);
        adj[v].push_back(u);
    }

    void dfs(int u, int p) {
        dfn[u] = low[u] = ++cur;
        stk.push_back(u);
        int child = 0;
        for(auto v : adj[u]) {
            if(!dfn[v]) {
                child++;
                dfs(v, u);
                low[u] = min(low[u], low[v]);
                if(low[v] >= dfn[u]) {
                    ++tot;
                    while(1) {
                        int now = stk.back();
                        stk.pop_back();
                        ecc[tot].push_back(now);
                        if(now == v) break;
                    }
                    ecc[tot].push_back(u);
                }
            } else if(v != p) {
                low[u] = min(low[u], dfn[v]);
            }
        }

        if(p == -1 && child == 0) {
            ++tot;
            ecc[tot].push_back(u);
        }
    }

    vector<vector<int>> work() {
        for(int i = 1; i <= n; ++i) {
            if(!dfn[i]) {
                dfs(i, -1);
            }
        }
        return ecc;
    }


```

```
};



void solve(){
    int n, m;
    cin >> n >> m;
    VBCC g(n);
    for(int i = 1; i <= m; i++) {
        int u, v;
        cin >> u >> v;
        g.addEdge(u, v);
    }

    auto ecc = g.work();
    int tot = g.tot;

    cout << tot << '\n';

    for(int i = 1; i <= tot; i++) {
        cout << ecc[i].size() << ' ';
        for(auto x : ecc[i]) cout << x << ' ';
        cout << '\n';
    }









}


int main(){
    std::ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);

    int T = 1;
    //cin >> T;
    while(T--) solve();

    return 0;
}
```

3.7 tarjan 求强连通分量

```
#include<bits/stdc++.h>
using namespace std;
using i64 = long long;
// 传n 使用下标 1~n
struct SCC {
    int n;
    vector<vector<int>> adj;
    vector<int> stk;
    vector<int> dfn, low, bel;
    // vector<vector<int>> ecc;
    int cur, tot;
    SCC(int n) {
        init(n);
    }

    void init(int n){
        this->n = n;
        adj.assign(n + 1, {});
        // ecc.assign(n + 1, {});
        dfn.assign(n + 1, 0);
        low.resize(n + 1);
        bel.assign(n + 1, 0);
        stk.clear();
        cur = tot = 0;
    }

    void addEdge(int u, int v) {
        adj[u].push_back(v);
    }

    void dfs(int u) {
        dfn[u] = low[u] = ++cur;
        stk.push_back(u);
        for(auto v : adj[u]) {
            if(!dfn[v]) {
                dfs(v);
                low[u] = min(low[u], low[v]);
            } else if(bel[v] == 0) {
                low[u] = min(low[u], low[v]);
            }
        }

        if(low[u] == dfn[u]) {
            ++tot;
            while(1) {
                int now = stk.back();
                // ecc[tot].push_back(now);
                bel[now] = tot;
                stk.pop_back();
                if(now == u) break;
```

```
            }
        }
    }

    vector<int> work() {
        for(int i = 1; i <= n; ++i) {
            if(!dfn[i]) {
                dfs(i);
            }
        }
        return bel;
    }


};



void solve(){

    int n, m;
    cin >> n >> m;

    vector<int> a(n + 1);
    vector<vector<int>> edge1(n + 1);
    for(int i = 1; i <= n; i++) cin >> a[i];
    SCC g(n);
    for(int i = 1; i <= m; i++) {
        int u, v;
        cin >> u >> v;
        g.addEdge(u, v);
        edge1[u].push_back(v);
    }

    auto bel = g.work();

    int tot = g.tot;
    vector<int> val(tot + 1), vis(tot + 1), ans(tot + 1);
    vector<vector<int>> edge2(tot + 1);

    for(int i = 1; i <= n; i++) {
        val[bel[i]] += a[i];
        for(auto j : edge1[i]) {
            if(bel[i] != bel[j]) edge2[bel[i]].push_back(bel[j]);
        }
    }
    int res = 0;
    auto dfs = [&] (auto self, int u, int p) -> void {
        if(vis[u]) return;
        vis[u] = p;
        for(auto v : edge2[u]) {
            if(vis[v] != u) {
                self(self, v, u);
                ans[u] = max(ans[u], ans[v]);
            }

        }
        ans[u] += val[u];
        res = max(res, ans[u]);
    };
    for(int i = 1; i <= tot; i++) {
        if(!vis[i]) {
            dfs(dfs, i, -1);
        }
    }
    cout << res << '\n';

}


int main(){
    std::ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);

    int T = 1;
    //cin >> T;
    while(T--) solve();

    return 0;
}
```

3.8 tarjan 求边双连通分量

```
#include<bits/stdc++.h>
using namespace std;
using i64 = long long;


struct EBCC {
    int n;
    vector<vector<int>> adj;
    vector<int> e;
    vector<int> stk;
    vector<int> dfn, low, bel;
    // vector<vector<int>> ecc;
    // vector<pair<int, int>> bridge; //割边
    int cur, tot;
```

```
    EBCC(int n) {
        init(n);
    }

    void init(int n){
        this->n = n;
        adj.assign(n + 1, {});
        // ecc.assign(n + 1, {});
        dfn.assign(n + 1, 0);
        low.resize(n + 1);
        bel.assign(n + 1, 0);
        stk.clear();
        e.clear();
        cur = tot = 0;
    }

    void addEdge(int u, int v) {
        adj[u].push_back(e.size());
        e.emplace_back(v);
        adj[v].push_back(e.size());
        e.emplace_back(u);
    }

    void dfs(int u, int laeid) {
        dfn[u] = low[u] = ++cur;
        stk.push_back(u);
        for(auto eid : adj[u]) {
            int v = e[eid];
            if(!dfn[v]) {
                dfs(v, eid);
                low[u] = min(low[u], low[v]);
            } else if(eid != (laeid ^ 1)) {
                low[u] = min(low[u], dfn[v]);
            }
        }

        if(low[u] == dfn[u]) {
            ++tot;
            // if(laeid != -1) bridge.push_back({u, e[laeid ^ 1]});
            while(1) {
                int now = stk.back();
                // ecc[tot].push_back(now);
                bel[now] = tot;
                stk.pop_back();
                if(now == u) break;
            }
        }
    }

    vector<int> work() {
        for(int i = 1; i <= n; ++i) {
            if(!dfn[i]) {
                dfs(i, -1);
            }
        }
        return bel; // bel : 每个点在哪个连通分量
    }


};
// 点下标为1~n是，传的参数n应该为n而不是n+1
// ecc可求出每个点在哪个连通分量 下标是 1~tot
// bel是每个点属于哪个连通分量 分量标号是1~tot
// tot是连通分量的个数
// bridge存的是每个割边的两个顶点



void solve(){
    int n, m;
    cin >> n >> m;
    EBCC g(n);

    for(int i = 1; i <= m; i++)  {
        int u, v;
        cin >> u >> v;
        g.addEdge(u, v);
    }

    auto bel = g.work();
    int tot = g.tot;
    cout << tot << '\n';

    vector<vector<int>> ebcc(tot + 1);

    for(int i = 1; i <= n; i++) {
        ebcc[bel[i]].push_back(i);
    }

    for(int i = 1; i <= tot; i++) {
        cout << ebcc[i].size() << ' ';
        for(auto it : ebcc[i]) cout << it << ' ';
            cout << '\n';
    }


```

```

}


int main(){
    std::ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);

    int T = 1;
    //cin >> T;
    while(T--) solve();

    return 0;
}
```

3.9 匈牙利算法

```
int match[N]; //match是表示右边的点匹配左边的哪一个点
set<int> st; // 每次dfs后要清空
bool dfs(int x) {
    for(auto it : edge[x]) {
        if(st.count(it)) continue;
        st.insert(it);
        if(!match[it] || dfs(match[it])) {
            match[it] = x;
            return 1;
        }
    }


    return 0;
}
```

3.10 最大流最小费用

```
template<class T>
struct MinCostFlow {
    struct _Edge {
        int to;
        T cap;
        T cost;
        _Edge(int to_, T cap_, T cost_) : to(to_), cap(cap_), cost(cost_) {}
    };
    int n;
    std::vector<_Edge> e;
    std::vector<std::vector<int>> g;
    std::vector<T> h, dis;
    std::vector<int> pre;
    bool dijkstra(int s, int t) {
        dis.assign(n, std::numeric_limits<T>::max());
        pre.assign(n, -1);
        std::priority_queue<std::pair<T, int>, std::vector<std::pair<T, int>>, std::
greater<std::pair<T, int>>> que;
        dis[s] = 0;
        que.emplace(0, s);
        while (!que.empty()) {
            T d = que.top().first;
            int u = que.top().second;
            que.pop();
            if (dis[u] != d) {
                continue;
            }
            for (int i : g[u]) {
                int v = e[i].to;
                T cap = e[i].cap;
                T cost = e[i].cost;
                if (cap > 0 && dis[v] > d + h[u] - h[v] + cost) {
                    dis[v] = d + h[u] - h[v] + cost;
                    pre[v] = i;
                    que.emplace(dis[v], v);
                }
            }
        }
        return dis[t] != std::numeric_limits<T>::max();
    }
    MinCostFlow() {}
    MinCostFlow(int n_) {
        init(n_);
    }
    void init(int n_) {
        n = n_;
        e.clear();
        g.assign(n, {});
    }
    void addEdge(int u, int v, T cap, T cost) {
        g[u].push_back(e.size());
        e.emplace_back(v, cap, cost);
        g[v].push_back(e.size());
        e.emplace_back(u, 0, -cost);
    }
    std::pair<T, T> flow(int s, int t) { //返回first: 最大流量, second: 最小费用
        T flow = 0;
        T cost = 0;
```

```
        h.assign(n, 0);
        while (dijkstra(s, t)) {
            for (int i = 0; i < n; ++i) {
                h[i] += dis[i];
            }
            T aug = std::numeric_limits<int>::max();
            for (int i = t; i != s; i = e[pre[i] ^ 1].to) {
                aug = std::min(aug, e[pre[i]].cap);
            }
            for (int i = t; i != s; i = e[pre[i] ^ 1].to) {
                e[pre[i]].cap -= aug;
                e[pre[i] ^ 1].cap += aug;
            }
            flow += aug;
            cost += aug * h[t];
        }
        return std::make_pair(flow, cost);
    }
    struct Edge {
        int from;
        int to;
        T cap;
        T cost;
        T flow;
    };
    std::vector<Edge> edges() {
        std::vector<Edge> a;
        for (int i = 0; i < e.size(); i += 2) {
            Edge x;
            x.from = e[i + 1].to;
            x.to = e[i].to;
            x.cap = e[i].cap + e[i + 1].cap;
            x.cost = e[i].cost;
            x.flow = e[i + 1].cap;
            a.push_back(x);
        }
        return a;
    }
};
```

4 字符串

4.1 KMP

```
int main()
{
    //P为模式串
    cin >> n >> p + 1 >> m >> s + 1;

    for (int i = 2, j = 0; i <= n; i ++ )
    {
        while (j && p[i] != p[j + 1]) j = ne[j];
        if (p[i] == p[j + 1]) j ++ ;
        ne[i] = j;
    }

    for (int i = 1, j = 0; i <= m; i ++ )
    {
        while (j && s[i] != p[j + 1]) j = ne[j];
        if (s[i] == p[j + 1]) j ++ ;
        if (j == n)
        {
            printf("%d ", i - n);
            j = ne[j];
        }
    }

    return 0;
}
```

4.2 hash string

```
#include <bits/stdc++.h>
#define uint uint64_t
#define int long long
using namespace std;

const uint mod = (1ull << 61) - 1;
mt19937_64 rnd(chrono::steady_clock::now().time_since_epoch().count());
uniform_int_distribution<uint> dist(mod / 2, mod - 1);
const uint base = dist(rnd);
const int N = 2e5 + 5;
vector<uint> p(N);

uint add(uint a, uint b)
{
    a += b;
    if (a >= mod)
        a -= mod;
    return a;
}

uint mul(uint a, uint b)
{
    __uint128_t c = __uint128_t(a) * b;
```

```
    return add(c >> 61, c & mod);
}

uint query(const vector<uint> &h, int l, int r)
{
    return add(h[r], mod - mul(h[l - 1], p[r - l + 1]));
}

void init()
{
    p[0] = 1;
    for (int i = 1; i < N; i++)
        p[i] = mul(p[i - 1], base);
}

template<typename T>
vector<uint> build(vector<T> &s)
{
    vector<uint> hashed(s.size(), 0);
    for (int i = 1; i < s.size(); i++)
        hashed[i] = add(mul(hashed[i - 1], base), s[i]);
    return hashed;
}

void solve()
{

}

signed main()
{
    // ios::sync_with_stdio(false);
    // cout.tie(nullptr);
    // cin.tie(nullptr);
    int T = 1;
    // cin >> T;
    while (T--)
        solve();
    return 0;
}
```

4.3 trie

```

struct Trie {

    int tot;
    vector<vector<int>> nex;
    vector<int> cnt; // 以这个节点结尾的字符串的个数
    Trie() : nex(100001, vector<int>(26)), cnt(100001) {}
    Trie(int n, int m) : nex(n + 1, vector<int>(m)), cnt(n + 1) {}
    // n为树中最多会有多少个节点，m表示有多少种字符
    void insert(string s, int t = 1) {    // 插入字符串
        int p = 0;
        for (int i = 0; i < s.length(); i++) {
            int c = s[i] - 'a';
            if (!nex[p][c]) nex[p][c] = ++tot;    // 如果没有，就添加结点
            p = nex[p][c];
        }
        cnt[p] += t;
    }

    bool find(string s) {    // 查找字符串
        int p = 0;
        for (int i = 0; i < s.length(); i++) {
            int c = s[i] - 'a';
            if (!nex[p][c]) return 0;
            p = nex[p][c];
        }
        return cnt[p];
    }
};
```

5 数学

5.1 exgcd

```
array<i64, 3> exgcd(i64 a, i64 b) {
    if(b == 0) {
        return {a, 1, 0};
    }
    auto [gd, x, y] = exgcd(b, a % b);
    return {gd, y, x - a / b * y};
}
```

5.2 millerRabin

```
//O(klog^3(n)), k = 7
#include <bits/stdc++.h>
using i64 = long long;

i64 qmi(i64 a, i64 b, i64 mod) {
    i64 res = 1;
    while(b) {
```

```
        if(b & 1) {
            res = (__int128)res * a % mod;
        }
        a = (__int128)a * a % mod;
        b >>= 1;
    }
    return res;
}

bool Minller(i64 n) {
    if(n <= 1 || n % 2 == 0) return (n == 2);
    i64 u = n - 1, k = 0;
    while(u % 2 == 0) u /= 2, ++k;
    static std::vector<i64> base = {2, 325, 9375, 28178, 450775, 9780504,
     1795265022};
    for(auto x : base) {
        i64 res = qmi(x, u, n);
        if(res == 0 || res == 1 || res == n - 1) continue;
        for(int i = 1; i <= k; ++i) {
            res = (__int128)res * res % n;
            if(res == n - 1) break;
            if(i == k) return false;
        }
    }
    return true;
}

void solve() {
    i64 x;
    std::cin >> x;
    std::cout << (Minller(x) ? "YES" : "NO") << '\n';
}

int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
    int T = 1;
    std::cin >> T;
    while(T--) {
        solve();
    }
    return 0;
}
```

5.3 qmi

```
int qmi(int a, int b) {
    int res = 1;
    while(b) {
        if(b & 1) res = 1ll * res * a % mod;
        b >>= 1;
        a = 1ll * a * a % mod;

    }
    return res;
}
```

5.4 扩展中国剩余定理

```
#include <bits/stdc++.h>
using namespace std;
using i64 = long long;

array<i64, 3> exgcd(i64 a, i64 b) {
    if(b == 0) {
        return {a, 1, 0};
    }
    auto [gd, x, y] = exgcd(b, a % b);
    return {gd, y, x - a / b * y};
}

i64 EXCRT(vector<pair<i64, i64>> &v) {
    i64 M = 1, ans = 0;

    for(const auto &[m, r] : v) {
        i64 c = ((r - ans) % m + m) % m;
        auto [gd, x, y] = exgcd(M, m);
        i64 x0 = (__int128)x * (c / gd) % (m / gd);
        i64 tmp = M * (m / gd);
        ans = (ans + (__int128)x0 * M % tmp) % tmp;
        M = tmp;
    }
    return (ans % M + M) % M;
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int n;
    cin >> n;
    vector<std::pair<i64, i64>> v(n);
    for(auto &[m, r] : v) {
        std::cin >> m >> r;
    }
    cout << EXCRT(v) << '\n';
    return 0;
```

```
}
```

5.5 欧拉筛 (和一个数质因数分解)

```
struct LinearSieve
{
    int n;
    vector<int> minp;
    vector<int> primes;
    vector<int> phi;
    vector<int> mu;

    LinearSieve(int n = 2e6 + 5) : n(n), minp(n + 1), phi(n + 1), mu(n + 1)
    {
        for (int i = 2; i <= n; i++)
        {
            if (minp[i] == 0)
            {
                minp[i] = i;
                phi[i] = i - 1;
                mu[i] = -1;
                primes.push_back(i);
            }
            for (int p : primes)
            {
                if (i * p >= n || p > minp[i])
                    break;

                minp[i * p] = p;

                if (p == minp[i])
                {
                    phi[i * p] = p * phi[i];
                    mu[i * p] = 0;
                    break;
                }
                else
                {
                    phi[i * p] = (p - 1) * phi[i];
                    mu[i * p] = -mu[i];
                }
            }
        }

        phi[1] = 1;
        mu[1] = 1;
    }

    map<int, int> factorize(int x) {
        map<int, int> facts;

        if (x <= n)
        {
            while (x > 1)
            {
                facts[minp[x]]++;
                x /= minp[x];
            }

            return facts;
        }

        for (int p : primes)
        {
            if (p * p > x) break;

            if (x % p == 0)
            {
                int count = 0;
                while (x % p == 0)
                {
                    x /= p;
                    count++;
                }
                facts[p] = count;
            }
        }

        if (x > 1) facts[x] = 1;

        return facts;
    }

    bool is_prime(int x)
    {
        if (x < 2 || x > n) return false;
        return minp[x] == x;
    }
}
```

5.6 矩阵

```
template<typename T>
struct Matrix {
    Matrix() : Matrix(0, 0) {};
    Matrix(int _n, int _m, T val = T{}) : n(_n), m(_m), mt(n * m, val){}
```

```
    Matrix(const std::initializer_list<std::initializer_list<T>> &v) : Matrix(v.size
    (), v.begin()->size()) {
        int i = 0;
        for(auto &row : v) {
            assert(row.size() == m);
            for(auto &x : row) {
                mt[i++] = x;
            }
        }
    }
    std::span<T> operator[](int idx) {
        return std::span<T>(mt.data() + idx * m, m);
    }
    std::span<const T> operator[](int idx) const {
        return std::span<const T>(mt.data() + idx * m, m);
    }
    friend Matrix<T> operator*(const Matrix<T>& A, const Matrix<T>& B) {
        assert(A.m == B.n);
        Matrix<T> MT(A.n, B.m);
        for(int i = 0; i < MT.n; ++i) {
            for(int j = 0; j < MT.m; ++j) {
                for(int k = 0; k < A.m; ++k) {
                    MT[i][j] = MT[i][j] + A[i][k] * B[k][j];
                }
            }
        }
        return MT;
    }
    friend Matrix<T> operator+(const Matrix<T>& A, const Matrix<T>& B) {
        assert(A.n == B.n && A.m == B.m);
        Matrix<T> MT(A.n, B.m);
        for(int i = 0; i < MT.n; ++i) {
            for(int j = 0; j < MT.m; ++j) {
                MT[i][j] = A[i][j] + B[i][j];
            }
        }
        return MT;
    }
    friend Matrix<T> operator-(const Matrix<T>& A, const Matrix<T>& B) {
        assert(A.n == B.n && A.m == B.m);
        Matrix<T> MT(A.n, B.m);
        for(int i = 0; i < MT.n; ++i) {
            for(int j = 0; j < MT.m; ++j) {
                MT[i][j] = A[i][j] - B[i][j];
            }
        }
        return MT;
    }
    static Matrix<T> eye(int n) {
        Matrix<T> MT(n, n);
        for(int i = 0; i < n; ++i) {
            MT[i][i] = 1;
        }
        return MT;
    }
    static Matrix<T> qpow(Matrix<T> a, i64 b) {
        Matrix<T> res(Matrix<T>::eye(a.n));
        while(b != 0) {
            if(b & 1) {
                res = res * a;
            }
            a = a * a;
            b >>= 1;
        }
        return res;
    }
    friend std::ostream& operator<<(std::ostream& os, const Matrix<T>& o) {
        for(int i = 0; i < o.n; ++i) {
            for(int j = 0; j < o.m; ++j) {
                os << o[i][j] << " \n"[j + 1 == o.m];
            }
        }
        return os;
    }
    int n, m;
    std::vector<T> mt;
};
```

5.7 线性筛

```
vector<int> minp, primes;

void sieve(int n) {
    minp.assign(n + 1, 0);
    primes.clear();

    for (int i = 2; i <= n; i++) {
        if (minp[i] == 0) {
            minp[i] = i;
            primes.push_back(i);
        }

        for (auto p : primes) {
            if (i * p > n) {
                break;
            }
```

```
            minp[i * p] = p;
            if (p == minp[i]) {
                break;
            }
        }
    }
}

bool isprime(int n) {
    return minp[n] == n;
}
```

5.8 组合数

```
vector<i64> fac, invfac;
int power(int a, int b) {
    int res = 1;
    while (b) {
        if (b % 2) res = 1LL * res * a % mod;
        a = 1LL * a * a % mod;
        b >>= 1;
    }
    return res;
}

void init(int n) {
    fac.resize(n + 1);
    invfac.resize(n + 1);

    fac[0] = 1;
    for (int i = 1; i <= n; i++) {
        fac[i] = 1LL * fac[i - 1] * i % mod;
    }
    invfac[n] = power(fac[n], mod - 2);  // 使用费马小定理求逆元
    for (int i = n - 1; i >= 0; i--) {
        invfac[i] = 1LL * invfac[i + 1] * (i + 1) % mod;
    }
}

// 计算组合数 C(n, m)
int binom(int n, int m) {
    if (n < m || m < 0) return 0;
    return 1LL * fac[n] * invfac[m] % mod * invfac[n - m] % mod;
}
int A(int n, int m) {
    return 1ll * fac[n] * invfac[n - m] % mod;
}
```

6 计算几何

6.1 凸包

```
#include <bits/stdc++.h>
using i64 = long long;
constexpr long double EPS = 1E-10;
const long double PI = acos(-1.0);
using T = long double;
struct Point {
    T x = 0, y = 0;
    Point operator+(const Point &o) const {return {x + o.x, y + o.y};}
    Point operator-(const Point &o) const {return {x - o.x, y - o.y};}
    Point operator-() const {return {-x, -y};}
    Point operator*(T fac) const {return {x * fac, y * fac};}
    Point operator/(T fac) const {return {x / fac, y / fac};}
    bool operator<(const Point &o) const {
        return std::tie(x, y) < std::tie(o.x, o.y);
    }
    friend std::istream &operator>>(std::istream &is, Point &p) {
        return is >> p.x >> p.y;
    }
    friend std::ostream &operator<<(std::ostream &os, Point p) {
        return os << "(" << p.x << ", " << p.y << ")";
    }
};

struct Line {
    Point s, t;
    Line() = default;
    Line(Point _s, Point _t) : s(_s), t(_t) {}
};

int sgn(T a){  //判断正负
    if(fabs(a) < EPS) return 0;
    return a > 0 ? 1 : -1;
}

T dot(const Point &a, const Point &b) { // 计算点积
    return a.x * b.x + a.y * b.y;
}
T cross(const Point &a, const Point &b) { // 两向量叉积
    return a.x * b.y - a.y * b.x;
}
T cross(const Point &a, const Point &b, const Point &c) { // ab向量与ac向量的叉积
    return cross(b - a, c - a);                           // c在直线ab左侧 > 0, 右侧 < 0, 共线 = 0
}
```

```cpp
T len(const Point &a) { // 求模长
    return sqrtl(a.x * a.x + a.y * a.y);
}
T angle(const Point &a, const Point &b) { // 求夹角
    return acosl(dot(a, b) / len(a) / len(b)); //(弧度制) [0, π]
    return acosl(dot(a, b) / len(a) / len(b)) * (180 / PI); // 转化为角度制 [0, 180]
}
T dis2(const Point &a, const Point &b) { // 距离的平方
    return (a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y);
}
T dis(const Point &a, const Point &b) { // 距离
    return sqrtl(dis2(a, b));
}
Point rotate(const Point &a, const Point &b, T theta) {  // 逆转角, ab向量逆时针旋转theta度, ac为转后向量,c为返回值
    return {
        (b.x - a.x) * cosl(theta) - (b.y - a.y) * sinl(theta) + a.x,
        (b.x - a.x) * sinl(theta) + (b.y - a.y) * cosl(theta) + a.y
    };
}

bool intersect(const Line &a, const Line &b) { //两线段相交返回1
    return cross(a.s, a.t, b.s) * cross(a.s, a.t, b.t) <= 0
        && cross(b.s, b.t, a.s) * cross(a.s, b.t, a.t) <= 0;
}
bool intersectStrictly(const Line &a, const Line &b) { // 两线段严格相交
    return cross(a.s, a.t, b.s) * cross(a.s, a.t, b.t) < 0
        && cross(b.s, b.t, a.s) * cross(a.s, b.t, a.t) < 0;
}

Point getNode(const Line &a, const Line &b) { // 两直线交点
    Point u = a.t - a.s, v = b.t - b.s;
    T t = cross(a.s - b.s, v) / cross(v, u);
    return a.s + u * t;

};

std::vector<Point> andrew(std::vector<Point> &v) {
    int n = v.size();
    std::sort(v.begin(), v.end());
    std::vector<Point> stk;
    for(int i = 0; i < n; ++i) {
        while(stk.size() > 1 && cross(stk[stk.size() - 2], stk.back(), v[i]) <= 0) {
            stk.pop_back();
        }
        stk.push_back(v[i]);
    }
    int t = stk.size();
    for(int i = n - 2; i >= 0; --i) {
        while(stk.size() > t && cross(stk[stk.size() - 2], stk.back(), v[i]) <= 0) {
            stk.pop_back();
        }
        stk.push_back(v[i]);
    }
    stk.pop_back();
    return stk;
};

T diameter(const std::vector<Point> &v) {
    int n = v.size();
    T res = 0;
    for(int i = 0, j = 1; i < n; ++i) {
        while(sgn(cross(v[i], v[(i + 1) % n], v[j]) - cross(v[i], v[(i + 1) % n], v[(j + 1) % n])) <= 0) {
            j = (j + 1) % n;
        }
        res = std::max({res, dis(v[i], v[j]), dis(v[(i + 1) % n], v[j])});
    }
    return res;
}

T diameter2(const std::vector<Point> &v) {
    int n = v.size();
    T res = 0;
    for(int i = 0, j = 1; i < n; ++i) {
        while(sgn(cross(v[i], v[(i + 1) % n], v[j]) - cross(v[i], v[(i + 1) % n], v[(j + 1) % n])) <= 0) {
            j = (j + 1) % n;
        }
        res = std::max({res, dis2(v[i], v[j]), dis2(v[(i + 1) % n], v[j])});
    }
    return res;
}

T grith(const std::vector<Point> &convex) {
    long double ans = 0;
    for(int i = 0; i < convex.size(); ++i) {
        ans += dis(convex[i], convex[(i + 1) % convex.size()]);
    }
    return ans;
}

void solve() {
    int n, m;
    std::cin >> n;
    std::vector<Point> A(n);
```

```
    for(int i = 0; i < n; ++i) {
        std::cin >> A[i];
    }
    std::cin >> m;
    std::vector<Point> B(m);
    for(int i = 0; i < m; ++i) {
        std::cin >> B[i];
    }
    long double ans = grith(A) + 2.0L * sqrtl(diameter2(B)) * acosl(-1.0L);   //A周长 + 2 * B直径 * PI
    std::cout << std::fixed << std::setprecision(15) << ans << '\n';
}

int main(){
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
    int T = 1;
    std::cin >> T;
    while(T--) {
        solve();
    }
    return 0;
}
```

7 杂项

7.1 常数表

n	$\log_{10} n$	$n!$	$C(n, n/2)$	LCM$(1...n)$	P_n
2	0.30102999	2	2	2	2
3	0.47712125	6	3	6	3
4	0.60205999	24	6	12	5
5	0.69897000	120	10	60	7
6	0.77815125	720	20	60	11
7	0.84509804	5040	35	420	15
8	0.90308998	40320	70	840	22
9	0.95424251	362880	126	2520	30
10	1.00000000	3628800	252	2520	42
11	1.04139269	39916800	462	27720	56
12	1.07918125	479001600	924	27720	77
15	1.17609126	1.31e12	6435	360360	176
20	1.30103000	2.43e18	184756	232792560	627
25	1.39794001	1.55e25	5200300	26771144400	1958
30	1.47712125	2.65e32	155117520	1.444e14	5604
P_n	37338_{40}	204226_{50}	966467_{60}	190569292_{100}	$1e9_{114}$

$\max \omega(n)$：小于等于 n 中的数最大质因数个数
$\max d(n)$：小于等于 n 中的数最大因数个数
$\pi(n)$：小于等于 n 中的数最大互质数个数

$n \leq$	10	100	1e3	1e4	1e5	1e6
$\max \omega(n)$	2	3	4	5	6	7
$\max d(n)$	4	12	32	64	128	240
$\pi(n)$	4	25	168	1229	9592	78498
$n \leq$	1e7	1e8	1e9	1e10	1e11	1e12
$\max \omega(n)$	8	8	9	10	10	11
$\max d(n)$	448	768	1344	2304	4032	6720
$\pi(n)$	664579	5761455	5.08e7	4.55e8	4.12e9	3.7e10
$n \leq$	1e13	1e14	1e15	1e16	1e17	1e18
$\max \omega(n)$	12	12	13	13	14	15
$\max d(n)$	10752	17280	26880	41472	64512	103680
$\pi(n)$	Prime number theorem: $\pi(x) \sim \frac{x}{\log(x)}$					

7.2 随机素数

979345007 986854057502126921
935359631 949054338673679153
931936021 989518940305146613
984974633 972090414870546877
984858209 956380060632801307

7.3 互质的规律

比较常见的定义 1. 较大数是质数, 两个数互质

2. 较小数是质数, 较大数不是它的倍数, 两个数互质

3. 1 与其他数互质

4. 2 与奇数互质

一些推论 1. 两个相邻的自然数一定互质

2. 两个相邻的奇数一定互质

3. n 与 2n + 1 或 2n - 1 一定互质

求差判断法如果两个数相差不大，可先求出它们的差，再看差与其中较小数是否互质。如果互质，则原来两个数一定是互质数。如：194 和 201，先求出它们的差，201 - 194 = 7，因 7 和 194 互质，则 194 和 201 是互质数。相反也成立, 对较大数也成立

求商判断法用大数除以小数，如果除得的余数与其中较小数互质，则原来两个数是互质数。如：317 和 52，317÷52 = 6.....5，因余数 5 与 52 互质，则 317 和 52 是互质数。

7.4 常见错因

爆数据 (爆 int, 爆 longlong)
取 mod 没有取干净或者取 mod 时超范围
想不出题事算一下各种数据范围

7.5 数论基础

7.5.1 欧拉函数

7.5.1.1 定义

$\varphi(n)$ 为 $1 \sim n$ 中和 n 互质的数的个数。

7.5.1.2 性质

若 p 是质数：

- $\varphi(p) = p - 1$
- $\varphi(p^k) = (p-1)p^{k-1}$

积性函数：若 $\gcd(m,n) = 1$，则 $\varphi(mn) = \varphi(m)\varphi(n)$。

7.5.1.3 质数分解公式

设 $n = \prod_{i=1}^{s} p_i^{a_i}$，则

$$\varphi(n) = n \times \frac{p_1 - 1}{p_1} \times \cdots \times \frac{p_s - 1}{p_s} = n \prod_{i=1}^{s} \left(1 - \frac{1}{p_i}\right).$$

7.5.2 莫比乌斯函数

7.5.2.1 定义

$$\mu(n) = \begin{cases} 1, & n = 1, \\ 0, & n \text{ 含有平方因子}, \\ (-1)^k, & k \text{ 是} n \text{ 的不同质因子的个数}. \end{cases}$$

7.5.3 数论定理

7.5.3.1 费马小定理

若 p 为质数且 $\gcd(a,p) = 1$，则

$$a^{p-1} \equiv 1 \pmod{p}.$$

7.5.3.2 欧拉定理

若 $\gcd(a,n) = 1$，则

$$a^{\varphi(n)} \equiv 1 \pmod{n}.$$

7.5.3.3 扩展欧拉定理

$$a^b \equiv \begin{cases} a^b, & b < \varphi(n), \\ a^{b \bmod \varphi(n) + \varphi(n)}, & b \geq \varphi(n), \end{cases} \pmod{n}$$

常用于降幂。

7.5.3.4 威尔逊定理

$$(p-1)! \equiv -1 \pmod{p}$$

这是 p 为素数的充要条件。若 $p > 4$ 且为合数，则

$$(p-1)! \equiv 0 \pmod{p}.$$

7.6 算法

7.6.1 出现的错误检查

- 超出范围 (int, int64)
- 取模出现错误
- 图可能有重边或者自环

7.6.2 杂项（通用）

想不到的题考虑：

1. 二分，二分答案，三分
2. DP
3. 离线
4. 倒推
5. 倍增
6. 建图

分块、势能分析、矩阵加速。
一个二进制状态，枚举它的所有子集：

$$\text{for}(j = \text{status};\ j > 0;\ j = (j-1)\&\text{status})$$

中位数处理方法：

- 二分 $\Rightarrow$ 转成 $+1, -1$ 判断
- 动态维护：加入元素求中位数：对顶堆
- 动态加入与删除：对顶 multiset

7.6.3 DP

- 背包 DP
- 区间 DP
- DAG 图上的 DP
- 树形 DP
  - 直接 DFS
  - DFN 序（可求子树大小、判断节点是否在子树中）
  - 换根 DP
  - 基环树 DP
- 状压 DP
- 数位 DP

7.6.4 搜索

- DFS，BFS
- 有权图不适合 BFS；01-BFS 适用于边权 0/1
- 双向 BFS：左右各搜一半再合并

7.6.5 图论

拓扑排序

可处理：

- 是否有环
- 环的大小
- 链的长度

Tarjan 缩点

- 边双联通：任意两点有两条边不相交路径
- 点双联通：任意两点有两条内部点不相交路径

7.6.6 树

倍增 LCA

树的重心

定义（等价）：

1. 最大子树最小
2. 任意子树节点数 $\leq \frac{n}{2}$
3. 所有点到该点的距离和最小

性质：

- 最多两个重心，且相邻
- 加删叶子时重心最多移动一条边
- 合并两棵树，新重心在原重心路径上
- 正权树上最小距离和点为重心

树的直径

求法：

1. 两次 DFS（正权）
2. 树形 DP（任意权）

性质：

- 多条直径有共同公共部分
- 对任意节点，与其最远点集合中包含直径端点之一

树上差分

- 点差分
- 边差分

其他

换根 DP、重链剖分、树上启发式合并。

树上启发式合并特征：

- 无修改操作
- 可通过 DFS 收集信息一次性回答所有询问

7.6.7 数学

数论

- 费马小定理
- 欧拉定理 / 扩展欧拉定理
- 威尔逊定理
- 中国剩余定理
- 欧拉函数
- 莫比乌斯函数

组合数学

- 组合数（含卢卡斯定理）
- 隔板法：n 个相同元素分成 k 组
- 容斥原理
- 卡特兰数：

$$f(n) = C(2n, n) - C(2n, n-1) = \frac{C(2n, n)}{n+1}$$

质数与因数分解

1. 小数：试除
2. 大数：Miller–Rabin
3. 区间多质数：欧拉筛 / 二次筛

质因数分解：

- 少量：试除 $O(\sqrt{n})$
- 多量：欧拉筛 + minp

其他：矩阵快速幂、整除分块、线性基、卢卡斯定理等。

生成函数

普通生成函数：

$$F(x) = \sum_{i \geq 0} a_i x^i$$

指数生成函数：

$$F(x) = \sum_{i \geq 0} a_i \frac{x^i}{i!}$$

分别用于多重集的组合数、排列数问题。

7.6.8 字符串

- KMP
- 字符串哈希
- Trie

7.6.9 数据结构

- 链表、栈、队列
- 单调栈：求前驱/后继大/小元素
- 单调队列：滑动窗口最值
- 堆：对顶堆维护中位数
- 并查集
  - 扩展域并查集（敌友关系）
  - 带权并查集（维护距离/取模）
- 线段树 (lazy tag、扫描线、动态开点等)
- 树状数组（维护差分、kth、小于当前数数量等）
- 波纹疾走树（区间第 k 小、频次查询）

7.7 组合数学

排列组合常用公式

1. $A_n^m = \dfrac{n!}{(n-m)!}$
2. 圆排列：$Q_n^m = \dfrac{A_n^m}{m} = \dfrac{n!}{m \cdot (n-m)!}$
3. $\dbinom{n}{m} = \dfrac{A_n^m}{m!} = \dfrac{n!}{m!(n-m)!}$
4. $\dbinom{n}{m} = \dbinom{n}{n-m}$
5. $\dbinom{n}{m} = \dfrac{n}{m}\dbinom{n-1}{m-1}$
6. $\dbinom{n}{m} = \dbinom{n-1}{m} + \dbinom{n-1}{m-1}$
7. $\sum\limits_{i=0}^{n} \dbinom{n}{i} = 2^n$
8. $\sum\limits_{i=0}^{n} (-1)^i \dbinom{n}{i} = 0$
9. 范德蒙恒等式：$\sum\limits_{i=0}^{k} \dbinom{n}{i}\dbinom{m}{k-i} = \dbinom{n+m}{k}$
10. $\sum\limits_{i=0}^{n} \dbinom{n}{i}^2 = \dbinom{2n}{n}$
11. $\sum\limits_{i=0}^{n} i\dbinom{n}{i} = n2^{n-1}$
12. $\sum\limits_{i=0}^{n} i^2\dbinom{n}{i} = n(n+1)2^{n-2}$
13. 朱世杰恒等式：$\sum\limits_{i=0}^{n} \dbinom{i}{m} = \dbinom{n+1}{m+1}$
14. $\sum\limits_{i=0}^{m} \dbinom{n+i}{n} = \dbinom{n+m+1}{n+1}$
15. $\dbinom{n}{r}\dbinom{r}{k} = \dbinom{n}{k}\dbinom{n-k}{r-k}$
16. $\sum\limits_{i=0}^{n} \dbinom{n-i}{i} = F_{n+1}$，其中 F 是斐波那契数列
17. 李善兰恒等式：$\sum\limits_{j=0}^{k} \dbinom{k}{j}^2 \dbinom{n+2k-j}{2k} = \dbinom{n+k}{k}^2$
18. $\sum\limits_{i=0}^{n} \dbinom{n}{i} a^i b^{n-i} = (a+b)^n$
19. $\sum\limits_{i=0}^{n} \dbinom{n}{i} 2^i = 3^n$

20. $H(x) = \sum\limits_{i=0} \sum\limits_{j+k=i} f(j) \cdot g(k) x^i$

$$F(x) = \sum_{i=0} f(i)x^i$$
$$G(x) = \sum_{i=0} g(i)x^i$$
$$H(x) = F(x) \cdot G(x)$$

Min-Max容斥

对于满足**全序**关系并且其中元素满足可加减性的序列 $\{x_i\}$，设其长度为 n，并设 $S = \{1, 2, 3, \cdots, n\}$，则有：

$$\max_{i \in S} x_i = \sum_{T \subseteq S} (-1)^{|T|-1} \min_{j \in T} x_j$$
$$\min_{i \in S} x_i = \sum_{T \subseteq S} (-1)^{|T|-1} \max_{j \in T} x_j$$

对于期望上：

$$E(\max_{i \in S} x_i) = \sum_{T \subseteq S} (-1)^{|T|-1} E(\min_{j \in T} x_j)$$
$$E(\min_{i \in S} x_i) = \sum_{T \subseteq S} (-1)^{|T|-1} E(\max_{j \in T} x_j)$$

对于 kth 上：

$$\operatorname{kthmax}_{i \in S} x_i = \sum_{T \subseteq S} (-1)^{|T|-k} \binom{|T|-1}{k-1} \min_{j \in T} x_j$$
$$\operatorname{kthmin}_{i \in S} x_i = \sum_{T \subseteq S} (-1)^{|T|-k} \binom{|T|-1}{k-1} \max_{j \in T} x_j$$
$$E\left(\operatorname{kthmax}_{i \in S} x_i\right) = \sum_{T \subseteq S} (-1)^{|T|-k} \binom{|T|-1}{k-1} E\left(\min_{j \in T} x_j\right)$$
$$E\left(\operatorname{kthmin}_{i \in S} x_i\right) = \sum_{T \subseteq S} (-1)^{|T|-k} \binom{|T|-1}{k-1} E\left(\max_{j \in T} x_j\right)$$

对于 gcd 和 lcm 上：

$$\operatorname{lcm}_{i \in S} x_i = \prod_{T \subseteq S} (-1)^{|T|-1} \gcd_{j \in T} x_j$$
$$\gcd_{i \in S} x_i = \prod_{T \subseteq S} (-1)^{|T|-1} \operatorname{lcm}_{j \in T} x_j$$

错位排列

定义：

设 $(a_1, a_2, \ldots, a_n)$ 是 $\{1, 2, \ldots, n\}$ 的一个全排列，若对任意的 $i \in \{1, 2, \ldots, n\}$ 都有 $a_i \neq i$，则称 $(a_1, a_2, \ldots, a_n)$ 是 $\{1, 2, \ldots, n\}$ 的错位排列。

用 D_n 表示 $\{1, 2, \ldots, n\}$ 的错位排列的个数。

递推公式：

$$D_0 = 1, D_1 = 0, D_2 = 1$$
$$D_n = (n-1)(D_{n-1} + D_{n-2})$$

通项公式：

$$D_n = n! \sum_{i=0}^{n} \frac{(-1)^i}{i!}$$

自然幂，上升幂、下降幂

自然幂：m^n 称作 m 的 n 次方，公式表达如下：

$$m^n = \prod_{i=0}^{n-1} m$$

上升幂：记作 $m^{\overline{n}}$，公式表达如下：

$$m^{\overline{n}} = \prod_{i=0}^{n-1} (m+i) = \frac{(n+m-1)!}{(n-1)!}$$

下降幂：记作 $m^{\underline{n}}$，公式表达如下：

$$m^{\underline{n}} = \prod_{i=0}^{n-1} (m-i) = \frac{n!}{(n-m)!}$$

上升幂和下降幂的转换：

$$(-m)^{\overline{n}} = (-1)^n m^{\underline{n}}$$
$$(-m)^{\underline{n}} = (-1)^n m^{\overline{n}}$$

组合数表示成下降幂形式：

$$\binom{n}{m} = \frac{n!}{m!(n-m)!} = \frac{n^{\underline{m}}}{m!}$$

上指标反转：

$$\binom{n}{m} = (-1)^m \binom{m-n-1}{m}$$

广义牛顿二项式定理

$$(1+x)^n = \sum_{i=0}^{\infty} \binom{n}{i} x^i$$
$$\frac{1}{(1-x)^n} = \sum_{i=0}^{\infty} \binom{-n}{i} (-x)^i = \sum_{i=0}^{\infty} \binom{n+i-1}{n-1} x^i$$

常见例子：

1.
$$\begin{aligned}(1+x)^{\frac{1}{2}} &= 1 + \sum_{n=1}^{\infty} \frac{\frac{1}{2}\left(\frac{1}{2}-1\right)\cdots\left(\frac{1}{2}-n+1\right)}{n!} x^n \\ &= 1 + \frac{1}{2}x + \sum_{n=2}^{\infty} \frac{(-1)^{n-1} \cdot 1 \cdot 3 \cdots (2n-3)}{2^n \cdot n!} x^n \\ &= 1 + \frac{1}{2}x + \sum_{n=2}^{\infty} \frac{(-1)^{n-1}}{(2n-1)\cdot 4^n} \binom{2n}{n} x^n \\ &= \sum_{n=0}^{\infty} \frac{(-1)^{n-1}}{(2i-1)\cdot 4^n} \binom{2i}{n} x^n\end{aligned}$$

2.
$$\begin{aligned}(1-4x)^{\frac{1}{2}} &= \sum_{n=0}^{\infty} \frac{(-1)^{n-1}}{(2n-1)\cdot 4^n} \binom{2n}{n} (-4x)^n \\ &= \sum_{n=0}^{\infty} -\frac{1}{2n-1} \binom{2n}{n} x^n\end{aligned}$$

3.
$$\begin{aligned}(1-4x)^{-\frac{1}{2}} &= \sum_{n=0}^{\infty} \binom{-\frac{1}{2}}{n} (-4x)^n \\ &= \sum_{n=0}^{\infty} (-1)^n \frac{1 \cdot 3 \cdot 5 \cdots (2n-1)}{2^n n!} (-4x)^n \\ &= \sum_{n=0}^{\infty} (-1)^n \frac{(2n)!}{2^{2n}(n!)^2} (-4x)^n \\ &= \sum_{n=0}^{\infty} (-1)^n \frac{1}{4^n} \binom{2n}{n} (-4x)^n \\ &= \sum_{n=0}^{\infty} (-1)^n \frac{1}{4^n} \binom{2n}{n} (-1)^n 4^n x^n \\ &= \sum_{n=0}^{\infty} \binom{2n}{n} x^n\end{aligned}$$

二项式反演

$$f_n = \sum_{i=0}^{n} \binom{n}{i} g_i \iff g_n = \sum_{i=0}^{n} (-1)^{n-i} \binom{n}{i} f_i$$

证明：

$$
\begin{aligned}
\sum_{i=0}^{n}\binom{n}{i}(-1)^{n-i}f(i) &= \sum_{i=0}^{n}\binom{n}{i}(-1)^{n-i}\sum_{j=0}^{i}\binom{i}{j}g(j)\\
&= \sum_{i=0}^{n}\sum_{j=0}^{i}\binom{n}{i}\binom{i}{j}(-1)^{n-i}g(j)\\
&= \sum_{i=0}^{n}\sum_{j=0}^{i}\binom{n}{j}\binom{n-j}{i-j}(-1)^{n-i}g(j)\\
&= \sum_{j=0}^{n}\binom{n}{j}g(j)\sum_{i=j}^{n}\binom{n-j}{i-j}(-1)^{n-i}\\
&= \sum_{j=0}^{n}\binom{n}{j}g(j)\sum_{i=0}^{n-j}\binom{n-j}{i}(-1)^{n-j-i}\\
&= \sum_{j=0}^{n}\binom{n}{j}g(j)\cdot 0^{n-j}\\
&= g(n)
\end{aligned}
$$

斐波那契数列

斐波那契数列的定义如下：

$$F_0=0,F_1=1,F_n=F_{n-1}+F_{n-2}$$

卢卡斯数列的定义如下：

$$L_0=2,L_1=1,L_n=L_{n-1}+L_{n-2}$$

斐波那契数列的通项公式为：

$$F_n=\frac{\left(\frac{1+\sqrt{5}}{2}\right)^n-\left(\frac{1-\sqrt{5}}{2}\right)^n}{\sqrt{5}}$$

卢卡斯数列的通项公式为：

$$L_n=\left(\frac{1+\sqrt{5}}{2}\right)^n+\left(\frac{1-\sqrt{5}}{2}\right)^n$$

于是：

$$\frac{L_n+F_n\sqrt{5}}{2}=\left(\frac{1+\sqrt{5}}{2}\right)^n$$

因此有：

$$L_n^2-5F_n^2=-4$$

斐波那契数列的矩阵形式：

$$\begin{bmatrix}F_{n-1} & F_n\end{bmatrix}=\begin{bmatrix}F_{n-2} & F_{n-1}\end{bmatrix}\begin{bmatrix}0 & 1\\1 & 1\end{bmatrix}$$

设 $P=\begin{bmatrix}0 & 1\\1 & 1\end{bmatrix}$，可得：

$$\begin{bmatrix}F_n & F_{n+1}\end{bmatrix}=\begin{bmatrix}F_0 & F_1\end{bmatrix}P^n$$

快速倍增法：

$$
\begin{aligned}
F_{2k}&=F_k\left(2F_{k+1}-F_k\right)\\
F_{2k+1}&=F_{k+1}^2+F_k^2
\end{aligned}
$$

斐波那契数列的性质：

1. $\sum_{i=1}^{n}F_i=F_{n+2}-1$
2. $\sum_{i=1}^{n}F_{2i-1}=F_{2n}$
3. $\sum_{i=1}^{n}F_{2i}=F_{2n+1}-1$
4. $\sum_{i=1}^{n}F_i^2=F_nF_{n+1}$
5. $F_{n+m}=F_{n+1}F_m+F_nF_{m-1}$
6. $F_{n-1}F_{n+1}-F_n^2=(-1)^n$
7. $F_{2n-1}=F_n^2+F_{n-1}^2$
8. $F_n=\frac{F_{n+2}+F_{n-2}}{3}$
9. $F_{2n}=F_n\left(F_{n+1}+F_{n-1}\right)$
10. $\forall k\in\mathbb{N},F_n|F_{nk}$
11. $\forall F_a|F_b,a|b$
12. $\gcd\left(F_n,F_m\right)=F_{\gcd(n,m)}$

卡特兰数

$Catalan$ 数列 H_n 其对应的序列为：

H_0	H_1	H_2	H_3	H_4	H_5	H_6
1	1	2	5	14	42	132

卡特兰数的常见公式：

$$
\begin{aligned}
H_n&=\begin{cases}\sum_{i=1}^{n}H_{i-1}H_{n-i} & n\geq 2,n\in\mathbf{N}_+\\ 1 & n=0,1\end{cases}\\
H_n&=\frac{4n-2}{n+1}H_{n-1}\\
H_n&=\binom{2n}{n}-\binom{2n}{n-1}=\frac{\binom{2n}{n}}{n+1}
\end{aligned}
$$

卡特兰数的封闭形式：

$$H_n = \sum_{i=1}^{n} H_{i-1}H_{n-i} \quad n \geq 2$$

$$H(x) = 1 + xH^2(x)$$

$$H(x) = \frac{1-\sqrt{1-4x}}{2x} = \sum_{n=0}^{\infty} \frac{\binom{2n}{n}}{n+1}x^n$$

斯特林数

第二类斯特林数

第二类斯特林数（斯特林子集数）$\begin{Bmatrix} n \\ k \end{Bmatrix}$，也可记做 $S(n,k)$，表示将 n 个两两不同的元素，划分为 k 个互不区分的非空子集的方案数。递推式：

$$\begin{Bmatrix} n \\ k \end{Bmatrix} = \begin{Bmatrix} n-1 \\ k-1 \end{Bmatrix} + k\begin{Bmatrix} n-1 \\ k \end{Bmatrix}$$

边界是 $\begin{Bmatrix} n \\ 0 \end{Bmatrix} = [n=0]$。

通项公式：

$$\begin{Bmatrix} n \\ k \end{Bmatrix} = \frac{1}{k!}\sum_{i=0}^{k}(-1)^i\binom{k}{i}(k-i)^n = \sum_{i=0}^{k}\frac{(-1)^{k-i}i^n}{(k-i)!i!}$$

同一行第二类斯特林数的计算：

$$A(x) = \sum_{i=0}^{\infty}\frac{-1^i}{i!}x^i$$

$$B(x) = \sum_{i=0}^{\infty}\frac{i^n}{i!}x^i$$

$$C(x) = \sum_{i=0}^{\infty}\begin{Bmatrix} n \\ i \end{Bmatrix}x^i = A(x) * B(x)$$

同一列第二类斯特林数的计算：

暂无

第二类斯特林数的性质：

1. $\begin{Bmatrix} n \\ 0 \end{Bmatrix} = 0^n$
2. $\begin{Bmatrix} n \\ 1 \end{Bmatrix} = 1$
3. $\begin{Bmatrix} n \\ 2 \end{Bmatrix} = 2^{n-1} - 1$
4. $\begin{Bmatrix} n \\ 3 \end{Bmatrix} = \frac{3^{n-1}+1}{2} - 2^{n-1}$
5. $\begin{Bmatrix} n \\ n \end{Bmatrix} = 1$
6. $\begin{Bmatrix} n \\ n-1 \end{Bmatrix} = \binom{n}{2}$

7. $\begin{Bmatrix} n \\ n-2 \end{Bmatrix} = \binom{n}{3} + 3 \cdot \binom{n}{4}$
8. $\begin{Bmatrix} n \\ n-3 \end{Bmatrix} = \binom{n}{4} + 10 \cdot \binom{n}{5} + 15 \cdot \binom{n}{6}$
9. $\sum_{k=0}^{n}\begin{Bmatrix} n \\ k \end{Bmatrix} = B_n$，其中 B_n 是贝尔数
10. $\forall n < m, \sum_{k=0}^{m}(-1)^k\binom{m}{k}(m-k)^n = 0$
11. $\sum_{k=0}^{m}(-1)^k\binom{m}{k}(m-k)^m = m!$
12. $n^k = \sum_{i=0}^{k}\begin{Bmatrix} n \\ i \end{Bmatrix} \cdot i! \cdot \binom{n}{i}$

第一类斯特林数

第一类斯特林数（斯特林轮换数）$\begin{bmatrix} n \\ k \end{bmatrix}$，也可记做 $s(n,k)$，表示将 n 个两两不同的元素，划分为 k 个互不区分的非空轮换的方案数。

递推式：

$$\begin{bmatrix} n \\ k \end{bmatrix} = \begin{bmatrix} n-1 \\ k-1 \end{bmatrix} + (n-1)\begin{bmatrix} n-1 \\ k \end{bmatrix}$$

边界是 $\begin{bmatrix} n \\ 0 \end{bmatrix} = [n=0]$。

通项公式：

没有通项公式

第一类斯特林数的性质：

1. $\begin{bmatrix} n \\ 1 \end{bmatrix} = (n-1)!$
2. $\begin{bmatrix} n \\ 2 \end{bmatrix} = (n-1)!\sum_{i=1}^{n-1}\frac{1}{i}$
3. $\begin{bmatrix} n \\ n-1 \end{bmatrix} = \binom{n}{2}$
4. $\begin{bmatrix} n \\ n-2 \end{bmatrix} = 2 \cdot \binom{n}{3} + 3 * \binom{n}{4}$
5. $\sum_{i=0}^{n}\begin{bmatrix} n \\ i \end{bmatrix} = n!$

贝尔数

贝尔数 B_n 表示将 n 个互不相同的集合划分成若干个非空集合的方案数。

递推公式：

$$B_n = \sum_{i=0}^{n-1}\binom{n-1}{i}B_i$$

通项公式：

$$B_n = \sum_{i=0}^{n} \begin{Bmatrix} n \\ i \end{Bmatrix}$$

生成函数：

$$B(x) = e^{e^x - 1}$$

贝尔数的性质：

1. $\forall p \in \mathbb{P}, B_{p^m+n} = mB_n + B_{n+1}$
2. $Bell$ 数列模质数 p 意义下循环节长度为 $\dfrac{p^p - 1}{p - 1}$

分拆数

分拆

将自然数n写成递降正整数和的表示。

$n = x_1 + x_2 + \ldots + x_k, \quad x_1 \geq x_2 \geq \ldots \geq x_k \geq 1$

分拆数

分拆数：p_n 。自然数 n 的分拆方法数。

n	0	1	2	3	4	5	6	7	8
p_n	1	1	2	3	5	7	11	15	22

递推公式：

$$F_n = \begin{cases} 0 & n < 0 \\ 1 & n = 0 \\ \sum\limits_{k=1}^{\infty} (-1)^{k-1} (F_{n-\frac{k(3k-1)}{2}} + F_{n-\frac{k(3k+1)}{2}}) & n > 0 \end{cases}$$

按照递推公式求分拆数的时间复杂度：$O(n\sqrt{n})$。

生成函数：

$$F(x) = \prod_{n=1}^{\infty} \frac{1}{1 - x^n} = \frac{1}{\sum\limits_{k=1}^{\infty} (-1)^k (x^{k(3k-1)} + x^{k(3k+1)})}$$

k 部分拆数

k 部分拆数：将 n 分成恰有 k 个部分的分拆，称为k部分拆数，记作 $p(n, k)$。

$n = x_1 + x_2 + \ldots + x_k, \quad x_1 \geq x_2 \geq \ldots \geq x_k \geq 1$

k 部分拆数 $p(n, k)$ 同时也是下面方程的解数：

$n - k = x_1 + x_2 + \ldots + x_k, \quad x_1 \geq x_2 \geq \ldots \geq x_k \geq 0$

如果这个方程里面恰有 j 个部分非 0，则恰有 $p(n-k, j)$ 个解。因此有和式：

$$p(n, k) = \sum_{j=0}^{k} p(n-k, j)$$

由此可得递推式：

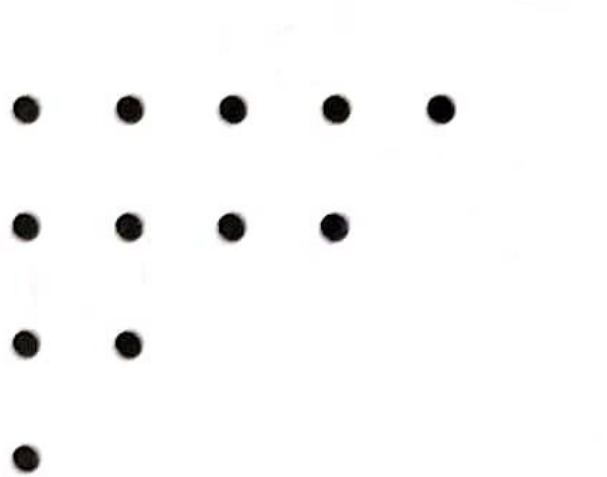
$$p(n, k) = p(n-1, k-1) + p(n-k, k)$$

Ferrers 图

Ferrers 图：将分拆的每个部分用点组成的行表示。每行点的个数为这个部分的大小。

根据分拆的定义，Ferrers 图中不同的行按照递减的次序排放。最长行在最上面。

例如：分拆 $12 = 5 + 4 + 2 + 1$ 的 Ferrers 图。



将一个 Ferrers 图沿着对角线翻转，得到的新 Ferrers 图称为原图的共轭，新分拆称为原分拆的共轭。显然，共轭是对称的关系。

例如上述分拆 $12 = 5 + 4 + 2 + 1$ 的共轭是分拆 $12 = 4 + 3 + 2 + 2 + 1$。

最大 k 分拆数

自然数 n 的分拆最大部分为 k 的分拆个数。

$n = x_1 + x_2 + \ldots + x_k, \quad x_1 \geq x_2 \geq \ldots \geq x_k \geq 1$，$x_1 = k$

根据共轭的定义，有显然结论：

最大 k 分拆数与 k 部分拆数相同，均为 $p(n, k)$。

互异分拆

将自然数n写成互异递降正整数和的表示。

$n = x_1 + x_2 + \ldots + x_k, \quad x_1 > x_2 > \ldots > x_k \geq 1$

互异分拆数

互异分拆数：pd_n。自然数 n 的各部分互不相同的分拆方法数。

n	0	1	2	3	4	5	6	7	8
p_n	1	1	1	2	2	3	4	5	6

k 部互异分拆数

k 部互异分拆数：将 n 分成恰有k个部分的分拆，称为k部分拆数，记作 $pd(n,k)$。

$n=x_1+x_2+\ldots+x_k,\quad x_1>x_2>\ldots>x_k\geq 1$

k 部分互异拆数 $pd(n,k)$ 同时也是下面方程的解数：

$n-k=x_1+x_2+\ldots+x_k,\quad x_1>x_2>\ldots>x_k\geq 0$

由于互异，新方程中至多只有一个部分为零。

因此直接得到递推：

$$pd(n,k)=pd(n-k,k-1)+pd(n-k,k)$$

普通生成函数

定义

序列 a 的普通生成函数（ordinary generating function，OGF）定义为形式幂级数：

$$F(x)=\sum_{n}^{\infty}a_nx^n$$

a 既可以是有穷序列，也可以是无穷序列。常见的例子（假设 a 以 0 为起点）：

1. 序列 $a=(1,2,3)$ 的普通生成函数是 $1+2x+3x^2$。
2. 序列 $a=(1,1,1,\cdots)$ 的普通生成函数是 $\sum\limits_{n=0}^{\infty}x^n$。
3. 序列 $a=(1,2,4,8,16,\cdots)$ 的生成函数是 $\sum\limits_{n=0}^{\infty}2^nx^n$。
4. 序列 $a=(1,3,5,7,9,\cdots)$ 的生成函数是 $\sum\limits_{n=0}^{\infty}(2n+1)x^n$。

换句话说，如果序列 a 有通项公式，那么它的普通生成函数的系数就是通项公式。

封闭形式

在运用生成函数的过程中，我们不会一直使用形式幂级数的形式，而会适时地转化为封闭形式以更好地化简。

例如 $1,1,1,\cdots$的普通生成函数 $F(x)=\sum\limits_{n=0}^{\infty}x^n$，我们可以发现

$$F(x)x+1=F(x)$$

那么解这个方程得到

$$F(x)=\frac{1}{1-x}$$

这就是 $\sum\limits_{n=0}^{\infty}x^n$的封闭形式。

考虑等比数列 $1,p,p^2,p^3,p^4,\cdots$ 的生成函数 $F(x)=\sum\limits_{n=0}p^nx^n$有

$$F(x)px+1=F(x)$$
$$F(x)=\frac{1}{1-px}$$

下列常见数列的普通生成函数（形式幂级数形式和封闭形式）。

1. $a=(0,1,1,1,1,\cdots)$。

$$F(x)=\sum_{n=1}^{\infty}x^n$$
$$F(x)=xF(x)+x$$
$$F(x)=\frac{x}{1-x}$$

2. $a=(1,0,1,0,1,\cdots)$。

$$F(x)=\sum_{n=0}^{\infty}x^{2n}$$
$$F(x)=x^2F(x)$$
$$F(x)=\frac{1}{1-x^2}$$

3. $a=(1,2,3,4,\cdots)$。

$$F(x)=\sum_{n=0}^{\infty}(n+1)x^n$$
$$F(x)=xF(x)+\sum_{n=0}^{\infty}x^n$$
$$F(x)=xF(x)+\frac{1}{1-x}$$
$$F(x)=\frac{1}{1-2x+x^2}=\frac{1}{(1-x)^2}$$

4. $a_n=\binom{m}{n}$（m是常数，$n\geq 0$）。

$$F(x)=\sum_{n=0}^{\infty}\binom{m}{n}x^n$$
$$F(x)=(1+x)^m$$

5. $a_n=\binom{m+n}{n}$（m是常数，$n\geq 0$）。

$$F(x)=\sum_{n=0}^{\infty}\binom{m+n}{n}x^n$$
$$F(x)=\frac{1}{(1-x)^{m+1}}$$

斐波那契数列的生成函数

斐波那契数列定义为 $a_0=0,a_1=1,a_n=a_{n-1}+a_{n-2}\ (n>1)$。

设它的普通生成函数是 $F(x)$，那么根据它的递推式，我们可以类似地列出关于 $F(x)$ 的方程：

$$F(x)=xF(x)+x^2F(x)+a_0+a_1x-a_0x$$

解得

$$F(x)=\frac{x}{1-x-x^2}$$

常见生成函数的封闭式

1. $\sum\limits_{i=0}^{n}x^{ki}=\dfrac{1-x^{kn+k}}{1-x^k}$
2. $\sum\limits_{i=0}^{n}\binom{n}{i}p^ix^i=(1+px)^n$
3. $\sum\limits_{i=0}^{\infty}x^{ki}=\dfrac{1}{1-x^k}$
4. $\sum\limits_{i=0}^{\infty}p^ix^i=\dfrac{1}{1-px}$
5. $\sum\limits_{i=1}^{\infty}x^i=\dfrac{x}{1-x}$
6. $\sum\limits_{i=0}^{\infty}(i+1)x^i=\dfrac{1}{(1-x)^2}$
7. $\sum\limits_{i=0}^{\infty}\binom{n+i}{i}x^i=\dfrac{1}{(1-x)^{n+1}}$

指数生成函数

定义

序列 a 的指数生成函数（exponential generating function，EGF）定义为形式幂级数：

$$\hat{F}(x)=\sum_{n=0}^{\infty}a_n\frac{x^n}{n!}$$

基本运算

$$\begin{aligned}\hat{F}(x)\hat{G}(x)&=\sum_{i=0}^{\infty}a_i\frac{x^i}{i!}\sum_{j=0}^{\infty}b_j\frac{x^j}{j!}\\&=\sum_{i=0}^{\infty}\sum_{j=0}^{i}a_jb_{i-j}\frac{1}{j!(i-j)!}x^i\\&=\sum_{i=0}^{\infty}\sum_{j=0}^{i}\binom{i}{j}a_jb_{i-j}\frac{x^i}{i!}\end{aligned}$$

常见指数生成函数的封闭式

1. $e^{px}=\sum\limits_{i=0}^{\infty}p^i\dfrac{x^i}{i!}$
2. $\dfrac{e^x+e^{-x}}{2}=\sum\limits_{i=0}^{\infty}\dfrac{x^{2i}}{(2i)!}$
3. $\dfrac{e^x-e^{-x}}{2}=\sum\limits_{i=0}^{\infty}\dfrac{x^{2i+1}}{(2i+1)!}$
4. $\begin{aligned}\ln(1+x)&=\sum_{i=1}^{\infty}(-1)^{i-1}(i-1)!\frac{x^i}{i!}\\&=\sum_{i=1}^{\infty}(-1)^{i-1}\frac{x^i}{i}\\&=x-\frac{x^2}{2}+\frac{x^3}{3}-\frac{x^4}{4}+\cdots\end{aligned}$
5. $\begin{aligned}-\ln(1-x)&=\sum_{i=1}^{\infty}(i-1)!\frac{x^i}{i!}\\&=\sum_{i=1}^{\infty}\frac{x^i}{i}\\&=x+\frac{x^2}{2}+\frac{x^3}{3}+\frac{x^4}{4}+\cdots\end{aligned}$

球盒模型

模型1

> n 个**无标号**的球放到 m 个**无标号**的盒子中，盒子**允许为空。**

即为 k 部分拆数 $p(n,m)$ 。

1. 直接递推

> 时间复杂度：$O(n^2)$

$$p(n,m)=p(n-1,m-1)+p(n-m,m)$$

模型2

> n 个**无标号**的球放到 m 个**无标号**的盒子中，盒子**至多装一个球。**

模型3

> n 个**无标号**的球放到 m 个**无标号**的盒子中，盒子**不允许为空。**

模型4

> n 个**无标号**的球放到 m 个**有标号**的盒子中，盒子**允许为空。**

模型5

> n 个**无标号**的球放到 m 个**有标号**的盒子中，盒子**至多装一个球**。

模型6

> n 个**无标号**的球放到 m 个**有标号**的盒子中，盒子**不允许为空**。

模型7

> n 个**有标号**的球放到 m 个**无标号**的盒子中，盒子**允许为空**。

模型8

> n 个**有标号**的球放到 m 个**无标号**的盒子中，盒子**至多装一个球**。

模型9

> n 个**有标号**的球放到 m 个**无标号**的盒子中，盒子**不允许为空**。

模型10

> n 个**有标号**的球放到 m 个**有标号**的盒子中，盒子**允许为空**。

模型11

> n 个**有标号**的球放到 m 个**有标号**的盒子中，盒子**至多装一个球**。

模型12

> n 个**有标号**的球放到 m 个**有标号**的盒子中，盒子**不允许为空**。

图论计数

外向树的拓扑序：

一棵 n 个点的外向树的拓扑序个数为：

$$n!\prod_{i=1}\frac{1}{size_i},size_i为节点i的子树大小$$

欧拉示性公式

对于任意的连通的平面图 G 有：

$$n-m+r=2$$

其中 n,m,r 分别为 G 的阶数，边数和面数。

式子化简

例一：

$$\sum_{i=0}^{n}\sum_{j=0}^{m}\binom{n}{i}\binom{m}{j}\left\lfloor\frac{|i-j|}{2}\right\rfloor\left(|i-j|-\left\lfloor\frac{|i-j|}{2}\right\rfloor\right)$$
$$=\sum_{i=0}^{n}\sum_{j=0}^{m}\binom{n}{i}\binom{m}{j}\left\lfloor\frac{|i-j|}{2}\right\rfloor\left\lceil\frac{|i-j|}{2}\right\rceil$$

1. 若 $|i-j|$ 为奇数：
   $$\left\lfloor\frac{|i-j|}{2}\right\rfloor\left\lceil\frac{|i-j|}{2}\right\rceil=\frac{(i-j)^2}{4}.$$
2. 若 $|i-j|$ 为偶数：
   $$\left\lfloor\frac{|i-j|}{2}\right\rfloor\left\lceil\frac{|i-j|}{2}\right\rceil=\frac{(i-j)^2-1}{4}.$$

$$\sum_{i=0}^{n}\sum_{j=0}^{m}\binom{n}{i}\binom{m}{j}\left\lfloor\frac{|i-j|}{2}\right\rfloor\left\lceil\frac{|i-j|}{2}\right\rceil$$
$$=\frac{1}{4}\left(\sum_{i=0}^{n}\sum_{j=0}^{m}\binom{n}{i}\binom{m}{j}(i-j)^2-\sum_{i=0}^{n}\sum_{\substack{j=0\\|i-j|\text{ is odd}}}^{m}\binom{n}{i}\binom{m}{j}\right)$$

$$\sum_{i=0}^{n}\sum_{j=0}^{m}\binom{n}{i}\binom{m}{j}(i-j)^2=\sum_{i=0}^{n}\sum_{j=0}^{m}\binom{n}{i}\binom{m}{j}(i^2-2ij+j^2)$$

$$\sum_{i=0}^{n}\sum_{j=0}^{m}\binom{n}{i}\binom{m}{j}i^2=\sum_{i=0}^{n}i^2\binom{n}{i}\sum_{j=0}^{m}\binom{m}{j}=n(n+1)2^{n+m-2}$$
$$\sum_{i=0}^{n}\sum_{j=0}^{m}\binom{n}{i}\binom{m}{j}ij=\sum_{i=0}^{n}i\binom{n}{i}\sum_{j=0}^{m}j\binom{m}{j}=nm2^{n+m-2}$$
$$\sum_{i=0}^{n}\sum_{j=0}^{m}\binom{n}{i}\binom{m}{j}j^2=\sum_{i=0}^{n}\binom{n}{i}\sum_{j=0}^{m}j^2\binom{m}{j}=m(m+1)2^{n+m-2}$$
$$\sum_{i=0}^{n}\sum_{\substack{j=0\\|i-j|\text{ is odd}}}^{m}\binom{n}{i}\binom{m}{j}=2^{n+m-1}$$

$$\sum_{i=0}^{n}\sum_{j=0}^{m}\binom{n}{i}\binom{m}{j}\left\lfloor\frac{|i-j|}{2}\right\rfloor\left\lceil\frac{|i-j|}{2}\right\rceil=((n-m)^2+n+m-2)2^{n+m-4}$$

例二：

$$F(x)=\frac{x}{1-x-x^2}$$

$$\frac{1}{1-F(x)}=\frac{1-x-x^2}{1-2x-x^2}=1+\frac{x}{1-2x-x^2}$$

$$\frac{A}{1-px}+\frac{B}{1-qx}=\frac{x}{1-2x-x^2}$$

$$=\frac{A+B-(Aq+Bp)x}{1-(p+q)x+pqx}$$

$$A+B=0$$

$$Aq+Bp=-1$$

$$p+q=2$$

$$pq=-1$$

$$p-q=2\sqrt{2}$$

$$p=1+\sqrt{2},q=1-\sqrt{2}$$

$$A=\frac{\sqrt{2}}{4},B=-\frac{\sqrt{2}}{4}$$

$$\frac{\sqrt{2}}{4}(\frac{1}{1-(1+\sqrt{2})x}-\frac{1}{1-(1-\sqrt{2})x})=\frac{\sqrt{2}}{4}\sum_{i=0}^{\infty}((1+\sqrt{2})^i-(1-\sqrt{2})^i)x^i$$

常用模数原根表

$p=r\times 2^k+1$

p	r	k	g
3	1	1	2
5	1	2	2
17	1	4	3
97	3	5	5
193	3	6	5
257	1	8	3
7681	15	9	17
12289	3	12	11
40961	5	13	3
65537	1	16	3
786433	3	18	10
5767169	11	19	3
7340033	7	20	3

p	r	k	g
23068673	11	21	3
104857601	25	22	3
167772161	5	25	3
469762049	7	26	3
754974721	45	23	11
998244353	119	23	3
1004535809	479	21	3
2013265921	15	27	31
2281701377	17	27	3
3221225473	3	30	5
75161927681	35	31	3
77309411329	9	33	7
206158430209	3	36	22
2061584302081	15	37	7
2748779069441	5	39	3
6597069766657	3	41	5
39582418599937	9	42	5
79164837199873	9	43	5
263882790666241	15	44	7
1231453023109121	35	45	3
1337006139375617	19	46	3
3799912185593857	27	47	5
4222124650659841	15	48	19
7881299347898369	7	50	6
31525197391593473	7	52	3
180143985094819841	5	55	6
1945555039024054273	27	56	5
4179340454199820289	29	57	3

$$F_n = m * (m-1)^{n-1} - (m-1)F_{n-2}$$

$$G(x) = \sum_{i=1} m * (m-1)^{i-1} x^i$$

$$G(x) = \frac{\frac{m}{m-1}}{1-(m-1)x} - \frac{m}{m-1}$$

$$F(x) = x + x^3$$

$$F(x) = G(x) - (m-1)F(x) * x^2$$

7.8 运行脚本

```
#!/bin/bash
g++ -std=c++20 -O2 -Wall "$1.cpp" -o "$1" -D_GLIBCXX_DEBUG
./"$1" < in.txt > out.txt
cat out.txt
```

7.9 VSCode 设置

- Auto Save (自动保存)
- Alt 键的快捷键
- Code Runner 运行快捷键
  - Run in Terminal
  - Run in Terminal
- 有时间可选: smooth